CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Unifying View on Timing Problems and Algorithms

Thibaut Vidal
Teodor Gabriel Crainic
Michel Gendreau
Christian Prins

July 2011

CIRRELT-2011-43

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

UNIVERSITÉ LAVAL · UQÀM · HEC MONTRÉAL · ÉCOLE POLYTECHNIQUE MONTRÉAL · Université de Montréal

A Unifying View on Timing Problems and Algorithms

Thibaut Vidal[1,2], Teodor Gabriel Crainic[1,3,*] Michel Gendreau[1,4], Christian Prins[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Laboratoire d'optimisation des systèmes industriels (LOSI), Université de Technologie de Troyes, 12, rue Marie Curie, B.P. 2060, 10010 Troyes, Cedex, France

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[4] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

Abstract. Timing problems involve the choice of task execution dates within a predetermined processing sequence, and under various additional constraints or objectives such as time-windows, time-dependent costs and so on. Their efficient solving is particularly critical in branch and bound and neighborhood search methods for vehicle routing, scheduling with idle time, as well as in various applications in network optimization or statistical inference. However, although timing related problems have been studied for many years, research on it suffers from a lack of common consensus, and most knowledge is scattered among operations research and applied mathematics domains. This article aims to introduce a classification for timing problems and features, as well as a unifying multidisciplinary analysis of timing algorithms. In relation to frequent application cases within branching schemes or neighborhood searches, efficient solving of series of similar timing subproblems is also analyzed. A dedicated formalism of re-optimization "by concatenation" is introduced to that extent, leading sometimes to considerable reductions in the computational burden of timing resolution. The knowledge developed through this analysis is valuable for modeling work and algorithmic design, for foremost problems such as rich vehicle routing variants, and many emerging non-regular scheduling applications.

Keywords: Timing, scheduling, routing, statistical inference, branch and bound, neighbourhood search.

Acknowledgements. While working on this project, T.G. Crainic was the Natural Sciences and Engineering Research Council of Canada (NSERC) Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway, M. Gendreau was the NSERC/Hydro-Québec Industrial Research Chair on the Stochastic Optimization of Electricity Generation, MAGI, École Polytechnique. and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal. Partial funding for this project has been provided by the Champagne-Ardennes regional council, France, the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by our partners CN, Rona, Alimentation Couche-Tard, la Fédération des producteurs de lait du Québec, the Ministry of Transportation of Québec, and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) through its Team Research Project program.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1 Introduction

Time-related constraints and objectives appear in a variety of flavors within scheduling, project management, data transmission, routing, network optimization and numerous other fields. Several problem settings in these domains involve the arrangement of elementary actions, named *activities*, under time requirements, such as tasks, visits to customers, production of objects, and so on. These activities come with unalterable features such as processing times, completion dates requirements, costs functions, and various other possible characteristics, while the decisional aspect generally involves choices on resource allocation (such as vehicles, machines, bandwidth) and execution dates. Most problems in the categories described previously can be seen as a merger of three issues (see Desrosiers et al. 1995): the repartition of activities among resources, called *resource allocation*, the choice of an execution order for activities on each resource, called *sequencing*, and finally the adjustment of activity execution dates with respect to this order, sometimes called *scheduling* or *timing*. In this paper we will favor the name *timing*, as the word *scheduling* is already employed for various settings in the literature.

In most situations, the combinatorial aspect of resource allocation and sequencing issues leads to NP-hard problem setting. Most heuristic searches and exact approaches perform a search through a large number of sequence and resource allocation alternatives, using repeatedly a timing algorithm to produce adequate execution dates, filter feasible solutions and evaluate costs. In all these cases the timing algorithm is called extensively, such that its complexity impacts dramatically the whole solution method, and makes the difference between successful algorithmic approaches and failure.

However, despite timing problems being the cornerstone of many algorithms, the literature on this subject remains scarce and scattered. Most developments on timing are made in relation to particular fields such as project planning, shortest path, routing, scheduling, as well as in other related mathematical disciplines like statistical inference that involve, quite unexpectedly, the same issues. Yet, few relationships between domains have been actually exploited, and thus close concepts and solution methods are independently developed within different formalisms, being rarely put in a more general context. The large variety of problem variants, arising from real life settings, also lead to considerable challenges regarding efficient timing solving. Such settings involve time-windows on activities (eventually multiple in Tricoire et al. 2010), penalized time-constraint transgressions (Taillard et al. 1997), time-dependent activity durations or costs (Hashimoto et al. 2008), speed decisions (Norstad et al. 2010), congestion or learning issues (Alidaee and Womer 1999, Kok et al. 2009), and so on. Although efficient timing algorithms have been designed for some of these characteristics taken separately, problems involving combinations of characteristics become much more complex, and there is generally no systematic way to derive concepts developed for the separate problems into a methodology for the new ones.

This paper contributes to the *timing field*, by means of a multidisciplinary review and analysis of timing features, problems and algorithmic approaches. A large assortment of problems, often treated independently in the literature under various names, are identified and classified in relation to their structure. Successful solution methods and solving concepts are reported and analyzed, and some brief new results are presented to fill most obvious methodological gaps. In the most noticeable cases, this analysis led to identify more than 26 algorithms from various research fields as similar implementations of three main generalist approaches (Sections 5.3-5.4). Not only does this review gather the keys for a stand-alone resolution of a large va-

riety of timing problems, but it also analyzes efficient timing solving as a subroutine in the context of global searches such as neighborhood-based heuristics and branch and bound based approaches. For these latter cases, managing global information through the successive solving of many similar timing instances can lead to dramatic reductions of the overall solving complexity. To this extent, a *re-optimization framework* is introduced in the second part of this paper. The body of knowledge developed in this paper is critical for both modeling work and algorithmic design, enabling to point out relationships between problems and their respective complexities. A portfolio of state-of-the-art timing algorithms is identified, which will prove useful to build more generalist solvers for many variants of difficult combinatorial optimization problems. To our knowledge, no such unifying review and methodological analysis of this rich body of problems has been performed in the past.

The remainder of this paper is organized as follows: Section 2 formally defines timing problems, while Section 3 presents examples of application contexts leading to such settings. Section 4 provides a detailed classification of the main timing features encountered in the literature as well as notations. Our methodological analysis of timing problems and their *independent resolution* is then organized in three Sections 5, 6 and 7 relatively to the previous classification. Section 8 finally introduces a *re-optimization framework* that encompasses state-of-the-art solving approaches for *solving series of similar timing instances*. Section 9 highlights some challenging avenues of research in the field of timing, and concludes our analysis.

2 Problem statement

In this paper, the term *activities* is used to represent, independently of the field of application, elementary operations that must be managed. The term *date* will always stand for a point in time, whereas the words *duration* or *time* will be employed for relative values (e.g., processing time). Without loss of generality, objective minimization will be considered.

Definition 1 (General timing problem and features). *Let $A = (a_1 \dots a_n)$ be a sequence of n activities with processing times $p_1 \dots p_n$. The execution dates of these activities $\mathbf{t} = (t_1 \dots t_n)$ constitute the decision variables of the timing problem, and are required to follow a total order with respect to the subscripts, such that $t_i \leq t_{i+1}$ for $i = 1 \dots n\text{-}1$.*

Other complicating features are also given. Let m be the number of such features. A feature F^x for $x \in [\![1, m]\!]$ involves the specification of additional problem parameters, along with a set of m_x characteristic functions $f_y^x(\mathbf{t})$ for each $y \in [\![1, m_x]\!]$ involving them. To each feature a role is also associated, either as objective or constraint. A feature either contributes to the objective with a value $\tilde{f}^x(\mathbf{t}) = \sum_{1 \leq y \leq m_x} f_y^x(\mathbf{t})$, or leads to a set of constraints $f_y^x(\mathbf{t}) \leq 0$ for $y \in [\![1, m_x]\!]$.

The general timing problem involves to state on the existence of a feasible timing solution $\mathbf{t} = (t_1 \dots t_n)$, respecting order and features constraints (for features $F^x \in \mathcal{F}^{\text{CONS}}$ having a role as constraint), and minimizing the weighted sum $c(\mathbf{t}) = \sum \alpha_x \tilde{f}^x(\mathbf{t})$ of contributions from all features $F^x \in \mathcal{F}^{\text{OBJ}}$ impacting the objective. This general problem is modeled in Equations (1-3):

$$\min_{\mathbf{t}=(t_1,..t_n)} \quad \sum_{F^x \in \mathcal{F}^{\text{OBJ}}} \alpha_x \tilde{f}^x(\mathbf{t}) \tag{1}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{2}$$

$$f_y^x(\mathbf{t}) \leq 0 \qquad F^x \in \mathcal{F}^{\text{CONS}}\ ,\ 1 \leq y \leq m_x \tag{3}$$

Timing problems can be viewed as shifting activity execution dates on a single resource, without never changing their processing order. This issue is equivalent to idle-time insertion when processing times are fixed. Most basic versions of timing are straightforward to solve, while the various features related to application cases can lead to dramatic increases in problem difficulty. Traditionally in the scheduling domain, some constraints and objectives are based on activity completion dates $C_i = t_i + p_i$. Without loss of generality, these expressions can be transformed in order to apply on execution dates. Finally, it must be noted that features have been defined independently from their role as constraint or objective, for two main reasons. First, many algorithms are concerned with the effective calculation of some quantities, like total duration for example, that enable to tackle related constraints or objectives in the same way. Furthermore, constraints can be transformed into objectives by Lagrangian relaxation, such that it is sometimes artificial to discriminate problems involving a given feature either as constraint or objective. Our study will thus be articulated around features in general, and their role will be specified only when relevant to the method.

Two illustrative examples of features follow. The feature *due date* D, for example, completes the model with $m_D = n$ additional parameters, representing latest execution dates d_i for activities, and involve the characteristic functions $f_i^D(\mathbf{t}) = (t_i - d_i)^+ \ (i = 1 \dots n)$. When D takes the role of a *constraint*, $f_i^D(\mathbf{t}) = (t_i - d_i)^+ \leq 0 \Leftrightarrow t_i \leq d_i$ yields the standard formulation of deadlines, while a role as objective leads to standard tardiness optimization criteria. The feature *time-lags* TL involves up to $m_{\text{TL}} = n^2$ additional parameters δ_{ij} on minimum elapsed time between the execution dates of activity pairs. The characteristic functions are $f_{ij}^{\text{TL}}(\mathbf{t}) = (t_j - \delta_{ij} - t_i)^+$. Depending on their role, these functions enable to model time-lag constraints, or minimize total infeasibility with respect to time-lags.

In order to emphasize the relations with practical problem settings, Section 3 details some major problems in the fields of operations research and applied mathematics leading to underlying timing structures.

3 Some application fields

Production scheduling and planning. The development of just-in-time policies in the industry, as well as the refinement of models, leads to challenging scheduling settings. Among others, time-dependent processing durations or costs (Alidaee and Womer 1999, Cheng 2004, Gawiejnowicz 2008), and scheduling with idle time or under non regular objectives are actually particularly active areas of research (Kanet and Sridharan 2000). In the earliness and tardiness (E/T) scheduling problem, we are given a sequence of activities $(a_1, \dots, a_n)$, their target execution dates d_i and processing time p_i, as well as penalty factors for earliness ϵ_i and tardiness τ_i. The goal is to determine the sequence of tasks and their execution dates on a single machine, such that linear penalties incurred for early or tardy processing are minimized (see Baker and Scudder 1990, for a review). This scheduling problem presents a non regular objective, and is NP-hard in the strong sense (Garey et al. 1988). Most recent approaches for (E/T) scheduling consider branch and bound, neighborhood search or other metaheuristics frameworks working on the sequence of activities. For each of the numerous sequences explored, a timing algorithm is applied to compute the activity execution dates and thus the sequence cost. This timing

problem is modeled in Equations (4-5).

$$\text{(E/T) timing:} \quad \min_{(t_1 \dots t_n)} \quad \sum_{i=1}^{n} \{\epsilon_i (d_i - t_i)^+ + \tau_i (t_i - d_i)^+\} \tag{4}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{5}$$

(E/T) timing appears as a particular case of the general timing formulation of Section 2 with release dates and deadlines (such that $r_i = d_i$ for $i = 1 \dots n$ to model target execution dates). The characteristic functions of these features, involved in the objective, are respectively $f_i(\mathbf{t}) = (r_i - t_i)^+$ and $f_i(\mathbf{t}) = (t_i - d_i)^+$. It is also noticeable that, in this case, different penalty coefficients are frequently associated to each customer and characteristic function. This timing problem is known to be solvable in $O(n \log n)$ (see Sections 5.3 and 5.4). Yet, as the efficiency of the timing procedure is the main bottleneck for most (E/T) scheduling approaches, extensive research has been conducted to solve series of timing instances more efficiently within neighborhood searches. The use of global information through the search leads to timing "re-optimization" procedures working in amortized $O(\log n)$ complexity, and even $O(1)$ for some particular cases, as described in Section 8.

Network optimization. Timing problems are also frequently encountered in various areas of network optimization, e.g. shortest path with resource constraints, vehicle scheduling, traveling salesman, vehicle routing with time features, and other complex settings (Solomon and Desrosiers 1988, Desrochers et al. 1990, Desrosiers et al. 1995). The vehicle routing problem with time-windows (VRPTW), in particular, consists in designing vehicle itineraries to service a set of geographically scattered customers within allowed time intervals. The VRPTW makes for one of the most intensively studied combinatorial optimization problem in the last decades, as underlined by dozens of literature reviews on the subject (see Kallehauge et al. 2005, Bräysy and Gendreau 2005a,b, Gendreau and Tarantilis 2010, for the most recent). Timing arises as a subproblem when checking the feasibility, or estimating the minimal amount of violation with respect to time-windows, on the sequences (itineraries) produced in the course of the search. Time-windows are a combination of both deadlines and release dates, and the resulting timing models are closely related to those presented previously in the case of (E/T) scheduling.

Break scheduling. Noteworthy are also several recent routing and crew scheduling applications, which require to explicitly consider regulations on working hours and breaks. Especially in long-haul vehicle routing, determining a feasible timing for a fixed sequence of visits makes for a highly complex problem. This problem is known to be solvable in $O(n^2)$ for United States working regulations (Archetti and Savelsbergh 2009, Goel and Kok 2009), while the existence of polynomial algorithms remains open in the case of European Regulations (Goel 2010, Prescott-Gagnon et al. 2010, Kok et al. 2010). These settings go beyond the timing formulations presented in Section 2, as they often necessitate to explicitly determine both the execution date and the nature of working and rest periods (breaks, daily rest periods and weekly rest periods). Still, the variety of regulations around the world raises a rich emerging body of problems which share many common points with timing settings, and should be considered relatively to them.

Energy efficient transportation. Norstad et al. (2010) introduce a ship routing problem with convex speed optimization, which presents two interlaced issues: the design of a ship itinerary, and the optimization of port arrival dates and speed (PATSO) in order to save energy.

For a fixed sequence of visits, the latter subproblem is formulated in Equations (6-9):

$$\text{(PATSO):} \quad \min_{\mathbf{t},\mathbf{v}} \sum_{i=1}^{n-1} d_{i,i+1}\, c(v_{i,i+1}) \tag{6}$$

$$s.t. \quad t_i + p_i + d_{i,i+1}/v_{i,i+1} \leq t_{i+1} \qquad 1 \leq i \leq n-1 \tag{7}$$

$$e_i \leq t_i \leq l_i \qquad 1 \leq i \leq n \tag{8}$$

$$v_{min} \leq v_{i,i+1} \leq v_{max} \qquad 1 \leq i \leq n-1 \tag{9}$$

In the previous formulation, the decision variables are the travel speeds $v_{i,i+1}$ for each port-to-port leg $i \in [\![1, n\text{-}1]\!]$, and the arrival dates at ports $t_i, i \in [\![1, n]\!]$. The objective function involves the sum of the costs of each trip leg, where $c(v)$ represents the cost per mile as a convex and increasing function of speed, $d_{i,i+1}$ represents the leg distances, and p_i denotes processing times at ports. The remaining constraints (7-9) respectively ensure that port arrival and departure dates are consistent with leg speeds, that earlier (e_i) and later bounds (l_i) on arrival dates to nodes are respected, and finally that speeds are in a feasible range.

This problem can be reformulated to rely on arrival dates only, as in (10-12), defining an extended cost/speed function $\dot{c}(v)$ (Equation 13) to account for the fact that waiting times can be used in case of forbidden low speed values.

$$\text{(PATSO-2):} \quad \min_{\mathbf{t}} \sum_{i=1}^{n} d_{i,i+1}\dot{c}\left(\frac{d_{i,i+1}}{t_{i+1}-t_i}\right) \tag{10}$$

$$\text{s.t.} \quad t_i + p_i + d_{i,i+1}/v_{max} \leq t_{i+1} \qquad 1 \leq i \leq n-1 \tag{11}$$

$$e_i \leq t_i \leq l_i \qquad 1 \leq i \leq n \tag{12}$$

$$\text{with} \quad \dot{c}(v) = \begin{cases} c(v_{min}) & \text{if } v \leq v_{min} \\ c(v) & \text{otherwise} \end{cases} \tag{13}$$

In this latter model, Equation (11) ensures that each leg can be operated at a feasible speed, and that port time windows are respected (12). This model falls into the category of timing problems. It involves time-windows features characterized by functions $f_i(\mathbf{t}) = (t_i - l_i)^+ + (e_i - t_i)^+$ with a role as constraints, as well as flexible processing times characterized by functions $f_i(\mathbf{t}) = c_i(t_{i+1} - t_i)$ in the objective, such that $c_i(\Delta t_i) = d_{i,i+1}\dot{c}(d_{i,i+1}/\Delta t_i)$. Norstad et al. (2010) propose a Recursive Smoothing Algorithm to solve the previous timing problem with a worst case complexity of $O(n^2)$. Several other timing algorithms, including re-optimization procedures, exist for these settings (Sections 6.3 and 8.6).

Statistical Inference. The isotonic regression problem under a total order (IRC) constitutes an intensively studied particular case of our models. Given a vector $\mathbf{N} = (N_1 \dots N_n)$ of n real numbers, IRC seeks a vector of non-decreasing values $\mathbf{t} = (t_1 \dots t_n)$ *as close as possible* to $\mathbf{N}$ according to a distance metric $\| \ \|$ (generally the Euclidean distance), as in Equations (14-15):

$$\text{(IRC):} \quad \min_{\mathbf{t}=(t_1,..t_n)} \|\mathbf{t} - \mathbf{N}\| \tag{14}$$

$$t_i \leq t_{i+1} \qquad 1 \leq i < n \tag{15}$$

As underlined by the seminal books of (Barlow et al. 1972, Robertson et al. 1988), IRC is the key to perform many restricted maximum likelihood estimates in statistics, and is linked with

various applications such as image processing and data analysis. It appears here as a timing problem with separable convex costs, similar to those encountered when solving vehicle routing problems with time-windows or (E/T) scheduling settings.

Other applications. Timing models finally appear under more general mathematical formalisms, under the name of *projection onto order simplexes* in Grotzinger and Witzgall (1984), or as particular cases of several convex optimization problems with underlying network structures (Hochbaum 2002, Ahuja et al. 2003). We now develop a new classification, as well as notations for the main classes of timing features and problems.

4 Features classification and reductions

This section first introduces a classification of main timing features in the literature, and notations for the related problems. We then examine reduction relationships between features, and describe the outline of this analysis.

4.1 Classification and notations

The main features encountered in the literature are here classified in relation to the mathematical structure of the characteristic functions they involve. We rely to that extent on a *feature dimension* measure ξ, which illustrates the links that a feature creates between decision variables:

Definition 2 (Feature dimension). *The dimension $\xi(F^x)$ of a feature F^x corresponds to the maximum number of variables involved together in any characteristic function $f^x_y(\mathbf{t})$ for $y = 1 \dots m_x$.*

Table 1 presents the most common features in the literature relatively to their dimension ξ. The first column provides an abbreviation for each feature, which will be used later on for problem notations. The next columns describes the parameters, characteristic functions and dimensions of these features. Finally, we describe the most frequent roles of each feature in the literature.

Most of the features presented in Table 1 are usual in the scheduling or vehicle routing literature. Features D, C, and W, can be qualified as *regular*, as they involve non-decreasing characteristic functions $f^x_y(\mathbf{t})$. For these regular features, the set of *actives schedules "such that no operation can be made to start sooner by permissible left shifting"* (Giffler and Thompson 1960) is dominating. Solving the timing problem is then straightforward by means of a minimum idle time policy (Section 5.1). However, these regular features present notably different behaviors with regards to re-optimization, thus motivating a detailed study. Other features from Table 1 are non-regular, and lead to more complex timing problems, for which idle-time insertion eventually leads to improvements with respect to objective or constraint satisfaction.

Single and two-dimension features are directly linked to physical quantities, respectively execution dates and durations, and are thus frequently encountered in timing formulations. Three dimension features are more unusual, and constitute a major step towards high complexity. Indeed, any mathematical program can be reformulated with constraints and objectives separable in groups of three variables, and thus almost any problem on totally ordered variables could be viewed as a timing problem in that sense. A reasonable limit to define *"what a timing problem is"* is to consider, as in the present paper, only applications and problems presenting explicitly the aspect of an activity sequence.

Table 1: Classification of timing features and notations

Symbol	Parameters	Char. functions	ξ	Most frequent roles
C	max exec date t_{max}	$f(\mathbf{t}) = (t_n - t_{max})^+$	1	Deadline on last activity, lateness of last activity, makespan
W	weights w_i	$f_i(\mathbf{t}) = w_i t_i$	1	Weighted execution dates
D	due dates d_i	$f_i(\mathbf{t}) = (t_i - d_i)^+$	1	Deadlines constraints, tardiness
R	release dates r_i	$f_i(\mathbf{t}) = (r_i - t_i)^+$	1	Release dates constraints, earliness.
TW	time-windows $TW_i = [e_i, l_i]$	$f_i(\mathbf{t}) = (t_i - l_i)^+ + (e_i - t_i)^+$	1	Time-window constraints, soft time-windows.
MTW	multiple TW $MTW_i = \cup[e_{ik}, l_{ik}]$	$f_i(\mathbf{t}) = \min_k [(t_i - l_{ik})^+ + (e_{ik} - t_i)^+]$	1	Multiple time-window constraints
$\Sigma c_i^{CVX}(t_i)$	convex $c_i^{CVX}(t_i)$	$f_i(\mathbf{t}) = c_i^{CVX}(t_i)$	1	Separable convex objectives
$\Sigma c_i(t_i)$	general $c_i(t)$	$f_i(\mathbf{t}) = c_i(t_i)$	1	Separable objectives, time-dependent activity costs
DUR	total dur. δ_{max}	$f(\mathbf{t}) = (t_n - \delta_{max} - t_1)^+$	2	Duration constraints, min duration excess
NWT	no wait	$f_i(\mathbf{t}) = (t_{i+1} - t_i)^+$	2	Min idle time
IDL	idle time ι_i	$f_i(\mathbf{t}) = (t_{i+1} - \iota_i - t_i)^+$	2	Min idle time excess
$P(t)$	time-dependent proc. times $p_i(t_i)$	$f_i(\mathbf{t}) = (t_i + p_i(t_i) - t_{i+1})^+$	2	Min activities overlap
TL	time-lags δ_{ij}	$f_i(\mathbf{t}) = (t_j - \delta_{ij} - t_i)^+$	2	Min excess with respect to time-lags
$\Sigma c_i(\Delta t_i)$	general $c_i(t)$	$f_i(\mathbf{t}) = c_i(t_{i+1} - t_i)$	2	Separable function of durations between successive activities, flex. processing times
$\Sigma c_i(t_i, t_{i+1})$	general $c_i(t, t')$	$f_i(\mathbf{t}) = c_i(t_i, t_{i+1})$	2	Separable objective or constraints by successive pairs of variables
$\Sigma c_{ij}(t_i, t_j)$	general $c_{ij}(t, t')$	$f_{ij}(\mathbf{t}) = c_i(t_i, t_j)$	2	Separable objective or constraints by any pairs of variables
$c(\mathbf{t})$	general $c(t)$	$f(\mathbf{t}) = c(\mathbf{t})$	-	Any feature

We rely in the following on a notation specifying for each problem the features considered, as well as information regarding their role. Each problem will be noted as a two component string $\{O|C\}$, where O is a list of features involving the objective and C lists features that participate to constraints. Separating features in the field O with a comma indicates a weighted sum of objectives, whereas the sign $\cup$ is used for multi-objective problems and the sign $>$ indicates an order of priority. Specificities on parameters are finally reported in parenthesis after the feature symbol. For example, problems with common deadlines can be marked with $(d_i = d)$, null processing times as $(p_i = 0)$ and so on.

To illustrate, consider the problem of speed optimization of Section 3. This problem presents a separable and convex objective as a function of durations between successive activities, along with time-window constraints. It can thus be categorized as $\{\Sigma c^{CVX}(\Delta t)|TW\}$. The (E/T) timing problem presents linear penalties around a due date, which can be assimilated to simultaneous release dates and deadlines, thus leading to the notation $\{R, D(r_i = d_i)|\emptyset\}$. Finally, the vehicle routing literature involves problem settings with a hierarchical objective aiming first to minimize the amount of time-window violations, then duration excess, and finally time-lag

violations. Such a problem setting can thus be characterized as $\{TW > D > TL|\emptyset\}$.

4.2 Feature reductions.

In the course of this paper, reduction relationships are used to illustrate the level of generality and complexity of timing features. A rich body of polynomial reductions has been already developed in the scheduling literature. Most of timing problems, however, are polynomially solvable, and the use of polynomial reduction relationships leads to encompass most problems in the same class of equivalence. We thus seek stronger reduction properties to distinguish them. We also aim to build relationships between features instead of complete problems, leading to the following definition of features reductions:

Definition 3 (Reducibility among timing features). *A feature F is said to be reducible to feature F' if any timing problem $\mathcal{T}$ involving F and other features $\{F^1 \dots F^k\}$ admits a linear many-one reduction to a timing problem $\mathcal{T}'$ involving $F' \cup \{F^1 \dots F^k\}$.*

To illustrate, the feature *release dates* R is reducible to the feature *time-windows* TW, as any instance of a timing problem with release dates, occurring in objectives or constraints, can be linearly transformed in an instance of a timing problem with time windows, the ends of time-windows being set to an arbitrary large value. Many other reduction relationships involving the features of Table 1 are introduced in the course of this paper.

An overview of the feature reductions is given in Figure 1, where an arrow from feature F^i to F^j indicates that feature F^i can be reduced to F^j. Four different categories of features are delimited by different shades of gray. On the left, we present features involving at most one decision variable (the first part of Table 1) and separable costs. Progressing to the right, the next gray shade represents two-dimension features that involve only pairs of consecutive activities together, then features involving any pair of activities together, and finally other features. We also demarcate the area of "NP-hard" features, which alone are sufficient to lead to NP-hard timing problems.

The hierarchy of reductions presented in Figure 1 gives an indication on the level of generality of features. Some features, such as $\Sigma c_i^{CVX}(t_i)$, $\Sigma c_i(t_i)$, $\Sigma c_i(\Delta t_i)$, and TL generalize many other features while remaining polynomially solvable. An algorithm addressing such general features can tackle many problems, while specialized algorithms for simpler combinations of features are more efficient. Both specialized and general algorithms are critical for practical applications, and deserve a detailed study. We thus provide a methodological review ordered by increasing generality, of the main features, timing problems and solution methods in the literature. We start with the most simple cases of single-dimension features in Section 5, and follow with two-dimension features in Section 6 to conclude our analysis of stand-alone methods for timing in Section 7.

5 Single-dimension features

Problems with single-dimension features are analyzed according to their difficulty and generality. We start with simple regular features, follow with time-windows TW features, separable convex costs $\Sigma c_i^{CVX}(t_i)$ and, finally, general separable costs $\Sigma c_i(t_i)$. The latter encompass multiple time-windows MTW and generalize all problems in this category.

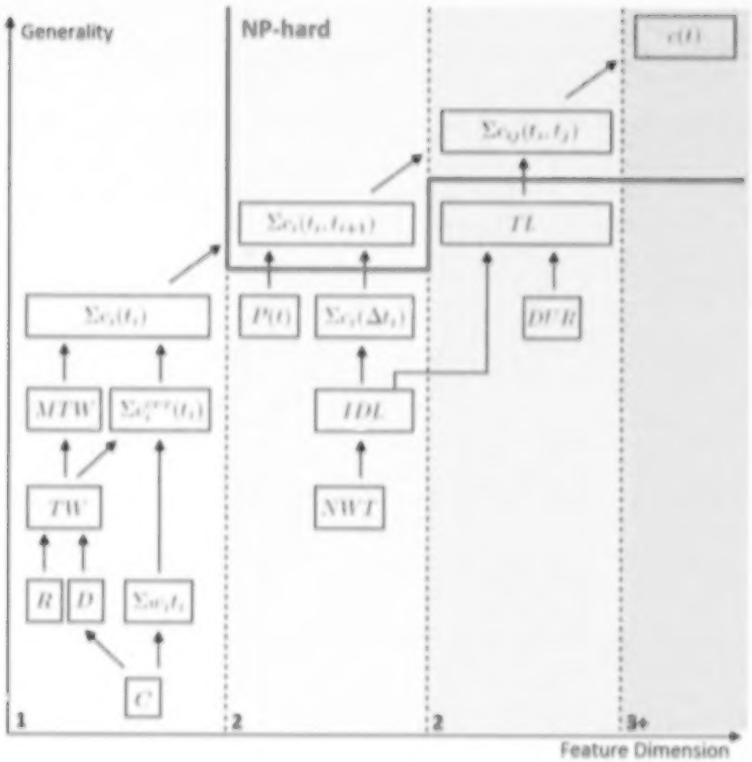


Figure 1: Hierarchy of timing features

5.1 Makespan, deadlines and weighted execution dates

Maximum execution date C, deadlines D, and weighted execution dates W features lead to many well-documented performance measures in the scheduling literature, and in operations research in general, such as tardiness, lateness and maximum completion dates (Graham et al. 1979, Pinedo 2008). W as an objective arises in various routing settings, such as the delivery-man problem (Fischetti et al. 1993), the minimum latency problem (Blum et al. 1994), and the cumulative TSP or VRP (Bianco et al. 1993, Ngueveu et al. 2010), where the goal is to service a set of customer as early as possible. These features are *regular* (see discussion in the previous Section), as any backward shift of execution date is beneficial for both feasibility and objective value. A very simple algorithm results from this observation, that will be refered to as the "minimum idle time policy": *For each activity a_i of A in the sequence order, schedule a_i at its earliest possible execution date. If a_i can not be scheduled, declare problem infeasibility and stop. If all activities have been scheduled successfully, declare problem feasibility.*

This simple way to choose execution dates is used extremely frequently in the literature to solve timing problems with regular features. An optimal solution is retrieved in n searches of the earliest feasible execution date, leading generally to $O(n)$ complexity.

5.2 Release dates and time-windows

Release dates and time-window features appear frequently in vehicle routing and scheduling applications. Time-window features generalize release dates R and deadlines D, as any release date r_i or deadline d_i can be transformed into a time-windows with an infinite value on the right side $[r_i, +\infty]$, or the left side $[-\infty, d_i - p_i]$.

Two main issues are often considered regarding these features. The first involves stating on the feasibility of a sequence of activities under time-window constraints, whereas the second problem involves the minimization of infeasibility with respect to the time-windows, and thus involve characteristic functions $f_i(\mathbf{t}) = (t_i - l_i)^+ + (e_i - t_i)^+$ in the objective.

Feasibility problem. Solving the feasibility problem $\{\emptyset|TW\}$ is straightforward, as the minimum idle time policy, presented in Section 5.1, is dominating in this respect. For a sequence of n activities $(a_1 \dots a_n)$, the algorithm starts with $t_1 = e_1$, then chooses each subsequent activity execution date in order to minimize idle time: $t_{i+1} = \max(t_i + p_i, r_{i+1})$. Hence, feasibility can be checked in $O(n)$. More efficient feasibility checking procedures have been developed by Savelsbergh (1985), in the context of local search for VRPTW, to solve *series of timing instances*. These procedures are presented in Section 8.

Minimize violation. Many real-case applications allow lateness or earliness, with respect to time-windows, as a way to gain flexibility in operations. The so-called *soft* time-window settings are very frequently used in heuristics for vehicle routing problems, as managing intermediate infeasible solutions contributes to improve the exploration capabilities of neighborhood searches, aiming to progress towards feasible solutions. Yet, different conventions for soft time-windows have been reported in the literature, that vary relative to earliness allowance and the way infeasibility is penalized. Several contributions, such as Taillard et al. (1997) and Cordeau et al. (2001), focus on the problem $\{D|R\}$, where late activities are allowed with penalties, but not early activities. This case falls within the scope of regular features (Section 5.1), and choosing for each activity the earliest execution date is optimal. The problem can thus be solved with linear complexity $O(n)$.

However, when early activities are allowed as in $\{TW|\emptyset\}$ (Koskosidis et al. 1992, Balakrishnan 1993, Ibaraki et al. 2005), the objective function is no longer non-decreasing. Supposing that activity a_i is finished earlier than the beginning of a_{i+1} time-window, choice must be made whether to insert idle time to reach a_{i+1}, or pay a penalty in order to better satisfy the time-windows of remaining activities. The resulting timing setting can thus become more complex. As an example, we show in Appendix A that the problem of minimizing the number of time-windows infeasibilities, that we notate $\{TW(unit)|\emptyset\}$, generalizes the Longest Increasing Subsequence Problem (LISP). LISP has been the subject of extensive research, and admits a computational lower bound of $\Omega(n \log n)$ in the comparison tree model (Fredman 1975).

It is remarkable that $\{TW|\emptyset\}$ is a special case of separable piecewise linear convex cost functions. Sections 5.3 and 5.4 will provide general efficient algorithms to tackle this wide range of problems, leading to an $O(n \log n)$ algorithm for soft time-window relaxations.

5.3 Separable convex costs

Separable convex costs Σc_i^{CVX} include a wide range of problem settings as particular cases. The feature TW, and thus R, D and C, can be reduced to $\Sigma c_i^{CVX}(t_i)$, as any time-window constraint can be modeled as a piecewise convex cost by associating arbitrary large costs to both sides of the feasibility interval. This feature also encompasses various other settings from the literature

such as earliness-tardiness scheduling with common or unequal due dates and various objective functions (Baker and Scudder 1990), isotonic regression problems with respect to a total order (Barlow et al. 1972, Robertson et al. 1988), extensions of team orienteering problems (also called traveling salesman with profits in Feillet et al. 2005), where eventually the profit value decreases with time (Erkut and Zhang 1996). It also includes problems with various convex penalty functions for time-window infeasibility (Sexton and Bodin 1985a,b, Ioachim et al. 1998, Ibaraki et al. 2005), or time-dependent convex processing costs (Tagmouti et al. 2007). The timing problem $\{\Sigma c_i^{CVX}(t_i)|\sigma\}$ is formulated in Equations (16-17).

$$\min_{(t_1 \dots t_n)} \quad \sum_{i=1}^{n} c_i^{CVX}(t_i) \tag{16}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{17}$$

The literature present a large choice of methods for this setting. We study, in this subsection, approaches specially designed for separable convex cost functions. Other dynamic programming based algorithms, relying on fundamentally different concepts, are grouped in Section 5.4.

To bring forth the main characteristics of timing algorithms for $\{\Sigma c_i^{CVX}(t_i)|\sigma\}$, we introduce a set of optimality conditions for the problem. The necessary and sufficient conditions we propose are more general than those developed previously in the literature (Best and Chakravarti 1990), being applicable to any set of proper convex cost functions, including non-smooth cases, which arise frequently in these settings. These conditions will be used to analyze the behavior of methods during the search.

Definition 4 (Activity blocks and prefix blocks). *A block B is defined as a subsequence of activities $(a_{B(1)} \dots a_{B(|B|)})$ processed consecutively (without idle time), such that $t_i + p_i = t_{i+1}$ for all $i \in \{B(1), \dots, B(|B|) - 1\}$. Let p_{ij} for $(i \leq j) \in [\![1, n]\!]^2$ be the cumulative processing duration of activities $a_i \dots a_j$. The block execution cost C_B as a function of its first activity execution time $t_{B(1)}$ is given in Equation (18).*

$$C_B(t_{B(1)}) = c_{B(1)}(t_{B(1)}) + \sum_{i=B(1)+1}^{B(|B|)} c_i(t_{B(1)} + p_{B(1)i-1}) \tag{18}$$

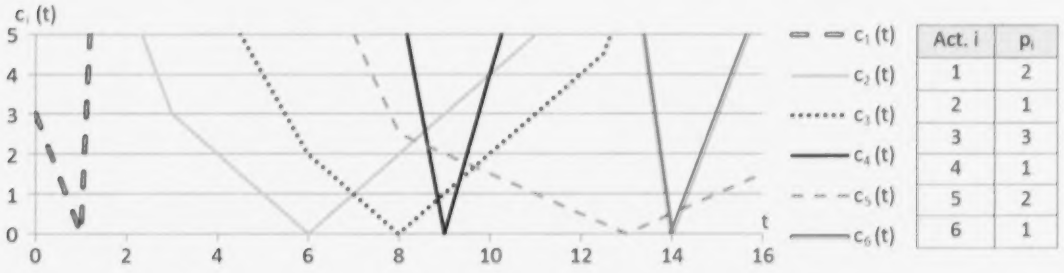
For any k such that $B(1) \leq k < B(|B|)$, we also define the sequence of activities $B^k = (a_{B(1)} \dots a_k)$ as a prefix block of B. Under the assumption that costs are proper convex functions (such that $\exists x | f(x) < +\infty$ and $\forall x, f(x) > -\infty$), the set of execution dates for the first activity minimizing this block execution cost is an interval, that we denote as $[T_B^{-*}, T_B^{+*}]$. These definitions enable to state the following necessary and sufficient optimality conditions:

Theorem 1. *Let costs $c_i(t_i)$ for $i = 1 \dots n$ be proper convex, eventually non-smooth, functions. A solution $\mathbf{t}^* = (t_1^* \dots t_n^*)$ of $\{\Sigma c_i^{CVX}(t_i)|\sigma\}$, assimilated to a succession of activity blocks $(B_1 \dots B_m)$, is optimal if and only if the three following conditions are satisfied:*

1. *Blocks are optimally placed: for each block B_i, $t^*_{B_i(1)} \in [T_{B_i}^{-*}, T_{B_i}^{+*}]$;*
2. *Blocks are strictly spaced: for each pair of blocks (B_i, B_{i+1}), $t^*_{B_i(1)} + p_{B_i(1)B_i(|B_i|)} < t^*_{B_{i+1}(1)}$;*
3. *Blocks are consistent: for each block B_i and prefix block B_i^k, $T_{B_i^k}^{+*} \geq t^*_{B_i(1)}$.*

Condition 2 can be stated as primal feasibility, while Condition 3 is related to dual feasibility.

Proof is given in Appendix B. We now analyze algorithms from the literature, and we distinguish two categories related to the respect of either these primal or dual conditions through the search. These algorithms have been published in totally different domains. In the case of isotonic regression in particular, only precedence constraints among decision variables are considered (and thus $p_i = 0$ for all i), yet these methods can generally be extended to solve problems with processing times with only minor modifications. Hence, we illustrate all these algorithms on a simple problem, for which the cost functions and the processing times are given in Figure 2.



Act. i	p_i
1	2
2	1
3	3
4	1
5	2
6	1

Figure 2: Illustrative example with six activities: cost functions and durations

5.3.1 Primal feasible and constructive methods:

In the literature, we can identify a first category of methods that always respect the primal feasibility conditions, and iteratively restore the dual conditions. The first algorithm of this type has been proposed by Brunk (1955) to solve isotonic regression problems. This so-called *Minimum Lower Set* (*MLS*) algorithm starts with a single big block, then iteratively for each block B, finds the biggest prefix block B^k violating dual conditions. If no such violation is found, this block is optimal, else the current block is split in two at this place, and the procedure is iterated on each sub-block until there is no remaining dual conditions violation. The algorithm can be implemented in $O(n^2)$ unimodal function minimizations.

Later on, Best and Chakravarti (1990) were the first to provide a primal feasible algorithm for IRC in $O(n)$ unimodal function minimizations. Again, activities are sequentially examined in each block to find the first violation of dual conditions (and not the most important violation). If such a violation exists, the block under consideration is split at this place. The leftmost resulting block having an earlier optimal starting date, it must eventually be merged with one or several previously scheduled blocks to reach an optimal execution date. When the cost functions are quadratic, analytic formulas exist to perform the unimodal function minimizations, such that the complexity of this algorithm becomes $O(n)$ (elementary operations).

In the domain of scheduling, Garey et al. (1988) propose a *timetabling algorithm* for (E/T) timing problems. The method iterates on activities in the order of the sequence, such that at step i the algorithm yields an optimal solution to the subproblem containing only the first i activities. Each new activity is inserted at the end of the schedule, to be then left-shifted and eventually merged into blocks with previous activities, until no improvement may be achieved. The algorithm runs in $O(n \log n)$ when relying on efficient data structures such as heaps.

Figure 3 illustrates the algorithms of Best and Chakravarti (1990) and Garey et al. (1988) on the example of Figure 2. The problem is solved in six steps, illustrated from top to bottom along

with the incumbent solutions, representing activities as rectangles with a length proportional to the processing time. We notice that the activity blocks in presence are very similar, such that these contributions can be seen as two different implementations of the same underlying primal feasible algorithm. The only difference is that Garey et al. (1988) consider non-inserted activities as non-existing in the current solution, whereas Best and Chakravarti (1990) keep those in one last block that does not respect the dual constraints.

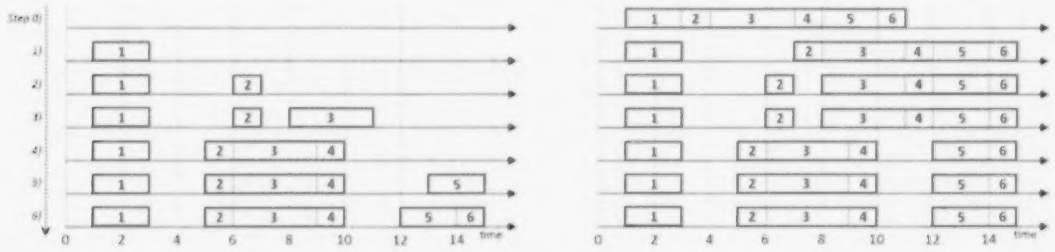


Figure 3: Comparison between Garey et al. (1988) algorithm (left part of the figure), and Best and Chakravarti (1990) algorithm (right part of the figure)

The method of Garey et al. (1988) has been extended by Lee and Choi (1995) and Pan and Shi (2005) to address timing problems in (E/T) scheduling with distinct penalty weights for earliness and tardiness $\{D, R(d_i - r_i)|\emptyset\}$ in $O(n \log n)$ elementary operations. Szwarc and Mukhopadhyay (1995) and Feng and Lau (2007) also propose to identify some tasks that are necessarily processed without idle time (in the same block) before solving. Chrétienne and Sourd (2003) apply the algorithm to project scheduling with general piecewise convex cost functions, and Hendel and Sourd (2007) to timing problems with convex piecewise linear cost functions or convex piecewise quadratic cost. These algorithms proceed in $O(n)$ unimodal function minimizations, but differ in terms of the data structures used to represent the functions and thus on the complexity of the function minimizations. When cost functions are piecewise linear, Hendel and Sourd (2007) attains a complexity of $O(\varphi_c \log n)$, where φ_c is the total number of pieces in all activity cost functions of the sequence.

We finally mention Davis and Kanet (1993), which propose another primal feasible method for (E/T) scheduling, generalized to general piecewise convex costs by Wan and Yen (2002). Activities are iteratively added to the solution in reverse order of the sequence. Each activity is scheduled at date 0, and then shifted onwards (while eventually merging blocks), until no improvement is achieved. This algorithm is equivalent to Garey et al. (1988) method on a symmetric problem with reversed sequence, under the change of variables $t'_i = M - t_i$, where M is a big time value.

5.3.2 Dual feasible methods.

Simultaneously with Brunk (1955), another seminal method for IRC was proposed by Ayer et al. (1955) under the name of *pool adjacent violators algorithm* (*PAV*). Starting with an initial solution consisting of n separate blocks, one for each activity, successive pairs of blocks (B_i, B_{i+1}) not satisfying primal conditions are iteratively identified. Such blocks are merged, and the next iteration is started. The order in which these couples of blocks are identified is a choice of implementation, and does not affect the final result of the algorithm. Figure 4

illustrates the PAV algorithm on the previous example. In this algorithmic implementation, we identify the first pair of blocks not verifying primal conditions to be merged. We notice that the optimal solution is reached after three merges (at Step 3 on the figure).

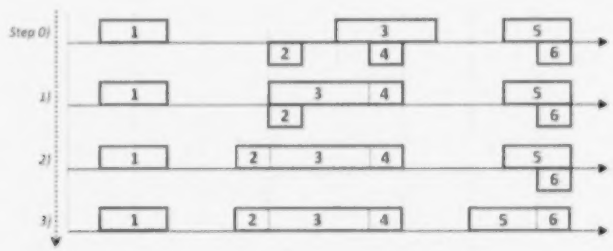


Figure 4: The PAV algorithm illustrated on timing problems

Chakravarti (1989) proved that PAV is a dual feasible method for the linear formulation of the problem, when the distance considered is $\| \ \|_1$ ($c(\mathbf{t}) = \Sigma|t_i - N_i|$), while Grotzinger and Witzgall (1984) and Best and Chakravarti (1990) showed that PAV is a dual algorithm for IRC when quadratic costs are considered (euclidean distance).

The PAV algorithm has also been generalized to convex functions by Best et al. (2000) and Ahuja and Orlin (2001), achieving a complexity of $O(n)$ unimodal minimizations. It is noteworthy that, under a totally different formalism, an equivalent algorithm has been discovered by Dumas et al. (1990) in the vehicle routing context with convex service costs as a function of time. For the case of the distance $\| \ \|_1$, the PAV algorithm can be implemented in $O(n \log^2 n)$ elementary operations using balanced search trees (Pardalos 1995), or $O(n \log n)$ complexity using scaling techniques (Ahuja and Orlin 2001). Finally, in the case of quadratic costs, analytic formulas are known for function minimas, such that $O(n)$ algorithms can be designed (Grotzinger and Witzgall 1984, Dumas et al. 1990, Pardalos and Xue 1999).

5.4 Separable costs and multiple time-windows

Without the previous convexity assumption, the timing problems $\{\Sigma c_i(t_i)|\emptyset\}$ become naturally more complex, and many authors have focused on separable piecewise linear costs. The feature MTW in particular (Christiansen and Fagerholt 2002, Tricoire et al. 2010), also appears as a case of piecewise linear separable cost, associating infinite cost to forbidden time periods.

When cost functions are non-negative and lower semi-continuous (l.s.c. : $c_i(t_i) \leq lim_{\epsilon \to 0} \min\{c_i(t_i + \epsilon), c(t_i - \epsilon)\}$ at every discontinuous point), dynamic programming concepts enable to solve efficiently the timing problems (Ibaraki et al. 2005). A large range of methods (Yano and Kim 1991, Sourd 2005, Ibaraki et al. 2005, Hendel and Sourd 2006, Ibaraki et al. 2008), tackling piecewise linear, and eventually convex costs, has been proposed in the routing and scheduling literature. Both backward or forward dynamic programming approaches have been applied.

Solving $\{\Sigma c_i(t_i)|\emptyset\}$ by forward dynamic programming involves the *forward minimum cost* function $F_i(t)$, which evaluates the minimum cost to execute the sequence of activities $(a_1 \dots a_i)$, while executing the last activity before t ($t_i \leq t$). $F_i(t)$ functions can be computed by means of Equation 19, starting from the case $i = 1$ with a single activity where $F_1(t) = \min_{x \leq t} c_1(x)$. The optimal solution value of the timing problem is $z^* = F_n(+\infty)$, and the optimal activity

execution dates are straightforward to retrieve from the functions $F_n(t)$.

$$F_i(t) = \min_{x \leq t}\{c_i(x) + F_{i-1}(x - p_{i-1})\} \qquad 1 < i \leq n \tag{19}$$

The symmetric way to solve this problem by backward programming involves the *backward minimum cost* function $B_i(t)$, which evaluates the minimum cost to execute the sequence of activities $(a_i \dots a_n)$, while executing the first activity a_i after t ($t_i \geq t$). $B_i(t)$ functions are computed by backward recursion, starting with $B_n(t) = \min_{x \geq t} c_n(x)$ and using Equation (20). After the calculation of $B_i(t)$ for all i, the optimal solution value of the timing problem is given by $z^* = B_1(-\infty)$.

$$B_i(t) = \min_{x \geq t}\{c_i(x) + B_{i+1}(x + p_i)\} \qquad 1 \leq i < n \tag{20}$$

Denoting φ_c the total number of pieces in all activity cost functions c_i, Ibaraki et al. (2005) provides an implementation of the forward or backward dynamic programming method working in $O(n\varphi_c)$. In the case where convex cost functions are considered, the use of efficient data structures such as balanced search trees leads to a complexity of $O(\varphi_c \log \varphi_c)$ (Ibaraki et al. 2008), matching the best available approaches in $O(n \log n)$ for the particular cases related to IRC, (E/T) scheduling or soft time-windows. Finally, besides their good complexity for solving independent timing problems, these dynamic programming approaches are suitable for efficient re-optimization procedures. Managing dynamic programming data through successive solving of similar timing instances, in neighborhood searches especially, can lead to a dramatic reduction of complexity. A thorough analysis of such *re-optimization procedures* for various features is provided in the second part of this article (Section 8).

5.5 State-of-the-art: single-dimension features

In this section we introduced and analyzed the main single-dimension features from the literature, independently of the field of application. The knowledge condensed here leads to a better understanding of the problems and algorithms at play, helpful for both modeling work and algorithmic design. For the problems considered, numerous contributions have been examined to identify state-of the art methods. In the particular case of $\{\Sigma c_i^{\text{cvx}}(t_i)|\emptyset\}$ and $\{\Sigma c_i(t_i)|\emptyset\}$, we classified 26 methods from various fields such as routing, scheduling and isotonic regression into three main families, that either respect dual or primal feasibility conditions on a suitable model, or exploit dynamic programming concepts. Another key result of this analysis is that a panel of three families of methods constitute the actual state-of-the-art for timing with single-dimension features. The simple *minimum idle time policy* enables to solve problems with regular features and objectives with a linear complexity. Primal and dual active sets methods solve timing problems with separable convex costs in n convex functions minimization. And finally, dynamic programming based methods are state of the art for separable (eventually convex) piecewise linear costs.

Single dimension features are related to many prominent problems such as LISP and IRC, which have already been the subject of extensive research. Still, we must keep in mind that only independent solving of timing problems has been thoroughly considered. As illustrated in Section 8, solving series of similar timing instances in local search context can be performed more efficiently by means of re-optimization procedures, and perspectives of research remain open in this area, even for single-dimension features.

6 Two-dimension features

We now focus on problems with two-dimension features. In most of the literature, these features are considered in presence of time-windows. TW is therefore often included as constraints in the models of this section. The presentation is structured in relation to the level of problem complexity and generality. Starting with the duration feature DUR, which involves exclusively the first activity and last activity together, we then examine two-dimension features involving successive activities: no-wait NWT, idle time IDL, and flexible $c_i(\Delta t_i)$ or time-dependent $P(t)$ processing times. Finally, features involving any pair of activities, such as time-lags TL, and cost functions separable by pairs of variables $\Sigma c_{ij}(t_i, t_j)$ are analyzed.

6.1 Total duration and total idle time

Accounting for total duration or idle-time is meaningful when one has the possibility to delay the beginning of operations; otherwise, considering the maximum execution date feature C would be sufficient. Whereas delaying the start of production is generally not an option in scheduling problems, it becomes particularly relevant in routing, as real-life objectives, relating to crew duty time, often involve duration minimization. We mention Savelsbergh (1992) for duration minimization under time-windows and duration constraints in VRPs, Cordeau et al. (2004) that generalizes the previous approach for soft time-windows and duration constraints, Desaulniers and Villeneuve (2000) for shortest path settings with linear idle-time costs and time windows, and Desaulniers et al. (1998) and Irnich (2008b) for a general framework that enables to tackle duration and idle time features, among others, in various time-constrained routing and crew scheduling problems.

Calculation of *total duration* and *total idle time* is equivalent, as under fixed processing times, the total duration and idle time of a schedule $\mathbf{t} = (t_1 \dots t_n)$ differ by one constant only (Equation 21).

$$WT(\mathbf{t}) = \sum_{i=1}^{n-1}(t_{i+1} - t_i - p_i) = t_n - t_1 - \sum_{i=1}^{n-1} p_i = DUR(\mathbf{t}) - \sum_{i=1}^{n-1} p_i \tag{21}$$

In order to manage duration features in $\{DUR|TW\}$ and $\{\emptyset|DUR, TW\}$, Savelsbergh (1992) proposes to first rely on a minimum idle time policy, and then shift activity execution dates forward to reduce the total duration. The related amount of shift has been introduced many years ago in the project scheduling literature (Malcolm et al. 1959), as the latest processing date for an activity which does not cause a slippage in the calendar duration. It is also known in the VRP literature under the name of *forward time slack* (Savelsbergh 1985, 1992).

The following quantities are computed for each activity a_i: the earliest feasible execution date T_i, the cumulative idle time W_i on the subsequence $(a_1 \dots a_i)$ according to these execution dates, and the partial forward time slack F_i on the subsequence $(a_1 \dots a_i)$. These values are recursively computed by means of Equations (22-24), starting with the single activity case for which $T_1 = e_1$, $W_1 = 0$ and $F_1 = l_1 - e_1$.

$$T_i = \max(T_{i-1} + p_{i-1}, e_i) \qquad 1 < i \leq n \tag{22}$$

$$W_i = W_{i-1} + T_i - T_{i-1} - p_{i-1} \qquad 1 < i \leq n \tag{23}$$

$$F_i = \min(F_{i-1}, l_i - T_i + W_i) \qquad 1 < i \leq n \tag{24}$$

The problem admits a feasible solution if and only if $T_i \leq l_i$ for all i. The execution date of the first activity in an optimal solution is given by $t_1^* = e_1 + \min\{F_n, W_n\}$. The other dates are computed using the minimum idle time policy. Both feasibility checking and duration minimization problems are thus solved in $O(n)$. Kindervater and Savelsbergh (1997a), Desaulniers and Villeneuve (2000) and Irnich (2008b) proposed different calculations of this optimal schedule. As pointed out by Parragh et al. (2010), all these approaches are closely related.

Tricoire et al. (2010) recently consider a more complex timing setting aiming to minimize duration under multiple time-window constraints $\{DUR|MTW\}$. Each activity a_i is now associated with a set of k_i time-windows, $MTW_i = \{[e_{i1}, l_{i1}] \dots [e_{ik_i}, l_{ik_i}]\}$. The authors propose a procedure that first removes some unnecessary time-windows segments, not suitable for any feasible solution, while detecting infeasible timing problems. In a second step, the procedure examines a subset of *dominant* schedules, such that "no better solution exists with the same last activity execution date". For a given execution date t_n of the last activity, a *dominant* schedule with minimum duration can easily be found using the backward recursion of Equation (25).

$$t_{i-1} = \max\{t \mid t \leq l_i - p_{i-1} \ \wedge \ t \in MTW_i\} \tag{25}$$

Starting from the dominant schedule $\bar{\mathbf{t}}$ with earliest completion time, the method iteratively identifies the last activity a_i followed by idle time: $t_i + p_i < t_{i+1}$. If activity a_i does not admit a later time-window, the algorithm terminates. Otherwise, the execution date of activity a_i is set to the beginning of the next time-window, and the execution dates of activities situated afterwards in the sequence are re-computed with a minimum idle time policy. The execution-time of the last activity obtained in this way leads to a *dominant* schedule, which becomes $\bar{\mathbf{t}}$ in the next iteration.

Tricoire et al. (2010) prove that at least one dominant schedule explored in the course of the algorithm is optimal. If each customer is associated to at least one time-window, the overall method can be implemented in $O(n\varphi_{\text{MTW}})$, where φ_{MTW} represent the number of time-windows in the problem.

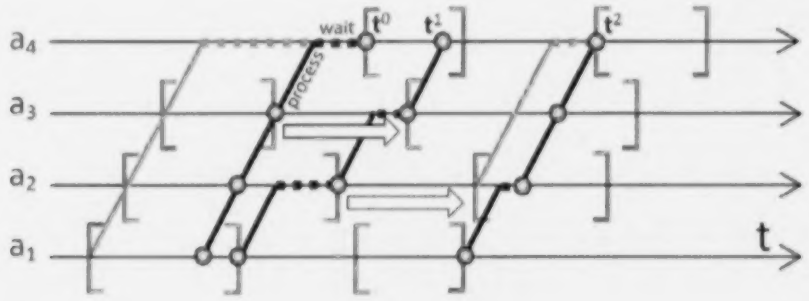


Figure 5: Duration minimization under multiple time-window constraints

We illustrate this algorithm in Figure 5 on a small example given with four activities and two time-windows per activity. Activities are represented from bottom to top with their time-windows, while the time dimension is represented by the horizontal axis. The earliest completion date is computed with a minimum idle time policy, illustrated in gray lines. The initial dominant schedule $\bar{\mathbf{t}}^0$, illustrated in black, is then determined by backward recursion using Equation 25.

This schedule presents waiting time after activity a_3, and thus the execution date of this activity is delayed to the next time-window, leading to a dominant schedule $\overline{\mathbf{t}}^1$. Now the latest activity followed by waiting-time is a_2. Its execution date is delayed, and leads to the dominant schedule $\overline{\mathbf{t}}^2$. The latest activity followed by waiting-time is now a_1, as there is no later time-window for this activity, the algorithm terminates. Among the three dominant schedules explored, the best solution with minimum duration has been reached by $\overline{\mathbf{t}}^2$, which is the optimal solution.

6.2 No-wait and idle time

No wait NWT and idle-time IDL features appear in various situations involving, among others, deterioration of products, maximum waiting times in passenger transportation, fermentation processes in the food industry, and cooling down in metal-casting processes. No-wait features are a special case of idle time IDL when $\iota_i = 0$. No-wait constraints $l_i = l_{i+1}$ can also be tackled by problem reformulation, merging unnecessary variables. When no waiting time is allowed on the entire activity sequence, the timing problem becomes a minimization problem of a sum of functions on one single variable.

Two main categories of problems have been considered in the literature for NWT and IDL features: the feasibility problem under idle-time and time-window constraints, presented here; and the optimization problem when some of these features appear in the objective function, treated in Section 6.3 in a more general context.

Feasibility checking under maximum idle time and time-window constraints has been frequently studied in the routing literature. Hunsaker and Savelsbergh (2002) design an algorithm to check the feasibility of itineraries in dial-a-ride settings, corresponding to $\{\emptyset|IDL, TW\}$ timing problem with additional time-lag features. This algorithm involves as special case a feasibility checking method for $\{\emptyset|IDL, TW\}$ working in $O(n)$. The solution to $\{\emptyset|IDL, TW\}$ is found in two scans of the activity sequence. The first pass considers the relaxed subproblem $\{\emptyset|TW\}$, determining for each activity i the earliest feasible execution dates T_i complying with time-windows and processing times. This calculation is performed by means of Equation (22). In a second pass, the algorithm proceeds backward in the sequence to determine the earliest feasible execution dates T'_i verifying also idle-time constraints (Equation 26), starting with $T'_n = T_n$. The problem is declared infeasible as soon as for any $i = 1 \ldots n$, $T_i > l_i$ or $T'_i > l_i$.

$$T'_i = \max(T'_{i+1} - p_i - \iota_i, T_i) \qquad 1 \leq i < n-1 \tag{26}$$

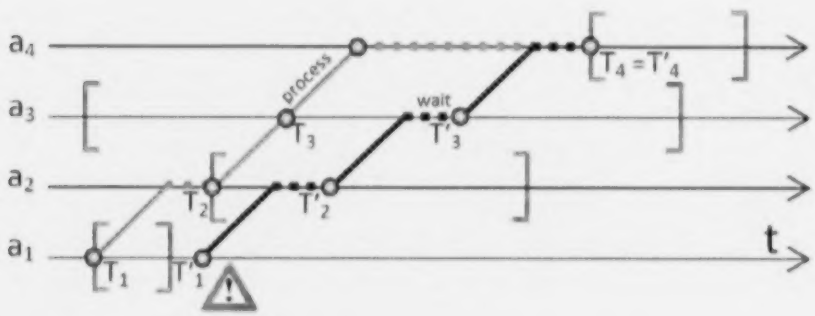


Figure 6: Feasibility checking under time-windows and idle-time constraints

The two passes are illustrated on Figure 6 for a small problem with four activities, represented from bottom to top with their time-windows. The first pass (T_i values) has been represented in gray, while the second backward scan, in black, provides the earliest feasible execution date for each activity $T_i^{\prime}$. As shown in the figure, this value exceeds the time-window of activity a_1 ($T_1^{\prime} > l_1$), and thus the timing problem illustrated is infeasible.

6.3 Flexible processing times

NWT, *IDL* features, and relaxed processing times are sometimes encountered in the objective function. In maritime transportation, Norstad et al. (2010) introduce a combined routing and speed optimization problem, where leg durations, assimilated to activity processing times, are a consequence of the speed choice (see Section 3). Time/resource trade-offs have also been studied for long in project scheduling, and shorter processing times for activities are eventually allowed with costs related to additional workforce, energy spent, or a different processing mode (Kelley and Walker 1959, Talbot 1982). The characteristic functions of *NWT* and *IDL* are also always separable by successive pairs of variables, such that these features reduce to $\Sigma c_i(\Delta t_i)$, often denoted as *flexible* processing times or travel times in routing literature.

If no further feature is added to the problem, $\{\Sigma c_i(\Delta t_i)|\emptyset\}$ is straightforward to solve, as all pairs of activities can be independently optimally spaced: $t_{i+1} - t_i = \arg\min_{\Delta t_i} \Sigma c_i(\Delta t_i)$.

In presence of other two-dimension features as constraints, on duration or idle-time, the problem $\{\Sigma c_i(\Delta t_i)|DUR\}$ can be reformulated into a continuous non-linear resource allocation problem, using the change of variables $x_i = t_{i+1} - t_i - p_i$. Numerous approaches for solving this wide class of problems have been proposed in the literature, as underlined by the remarkable review and annotated bibliography of Patriksson (2008).

The speed optimization problem of Norstad et al. (2010) (Equations 10-12) can be seen as a special case of $\{\Sigma c_i^{CVX}(\Delta t_i)|TW\}$, where the objective can be expressed as $z(t) = \sum_{i=1}^{n} d_{i,i+1} c\left(\frac{t_{i+1}-t_i}{d_{i,i+1}}\right)$, and functions $c(\Delta t)$ are non-increasing and convex. In presence of such functions, removing time-window constraints leads to optimal solutions presenting a constant ratio $\frac{t_{i+1}-t_i}{d_{i,i+1}}$ for all i, interpreted as a constant speed on all legs. The *recursive smoothing algorithm* (RSA) of Norstad et al. (2010) exploits this property, by maintaining this ratio constant on subsequences, while re-introducing progressively violated time-window constraints. It is remarkable that, if the minimum of $c(\Delta t)$ is given, the functions properties enable a resolution with no other call to the objective, and the algorithm complexity is $O(n^2)$ *elementary operations*.

In general settings though, when flexible processing times are combined with single-dimension features such as time-windows or time-dependent activity costs, the problem difficulty increases dramatically. Sourd (2005) and Hashimoto et al. (2006) have independently studied $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$ (Equation 27) in the context of (E/T) scheduling and vehicle routing. Both authors report the NP-hardness of this problem if no assumption is made on the functions at hand.

$$\min_{t_1 \ldots t_n} \quad \sum_{i=1}^{n} c_i(t_i) + \sum_{i=1}^{n-1} c_i^{\prime}(t_{i+1} - t_i) \tag{27}$$

In the case where the functions are piecewise linear with integer breakpoints, a dynamic programming algorithm is proposed to solve the problem in $O(T^2)$ where T represents an upper bound on the schedule durations. As in Section 5.4, this algorithm can be implemented with a forward dynamic programming function $F_i(t)$ (Equations 28-29), which now evaluates

the minimum cost to process the subsequence of activities $(a_1 \dots a_i)$, starting the last activity *exactly at date* t ($t_i = t$). The resulting optimal cost is given by $z^* = \min_t F_n(t)$.

$$F_1(t) = 0 \tag{28}$$

$$F_i(t) = c_i(t) + \min_{0 \le x \le t} \{F_{i-1}(x) + c'_{i-1}(t-x)\} \qquad 1 < i \le n \tag{29}$$

In the particular case where functions $c'_i(\Delta t)$ are piecewise linear and convex, Sourd (2005) and Hashimoto et al. (2006) propose a polynomial time implementation of the previous approach. Let φ_c and φ'_c be, respectively, the total number of pieces in cost functions c_i and c'_i, and $\widehat{\varphi_c}$ the number of convex pieces in the cost functions c_i. The resulting complexity is $O(n(\varphi_c + \widehat{\varphi_c} \times \varphi'_c))$. Efficient *re-optimization procedures* have also been developed by the authors (see Section 8).

Finally, DUR involved in the objective can be seen as a special case of $\Sigma c_i(\Delta t_i)$ where $c_i^{\text{DUR}}(\Delta t_i) = \Delta t_i$. MTW is also reducible to a piecewise linear $\Sigma c_i(t_i)$ with a total number of pieces proportional to $n + \varphi_{\text{MTW}}$, where φ_{MTW} represents the total number of time-windows. The previous algorithm thus provides an alternative way to solve $\{DUR|MTW\}$ or $\{\emptyset|DUR, MTW\}$ in $O(n + n\varphi_{\text{MTW}})$.

6.4 Time dependent processing times

In real life settings, activity processing times are often subject to variations. In scheduling for example, several studies on learning, deterioration effects and other time-dependencies on processing times have been performed (see the reviews from Alidaee and Womer 1999, Cheng 2004). In routing and data transmission, network congestion and rush hours are a major concern (Van Woensel et al. 2008, Kok et al. 2009), and thus the time-dependent processing-time feature $P(t)$ appears in various network optimization problems: shortest path (Cooke and Halsey 1966, Dreyfus 1969, Halpern 1977), traveling salesman (Malandraki and Dial 1996), vehicle routing (Beasley 1981, Malandraki and Daskin 1992, Ichoua et al. 2003, Donati et al. 2008, Hashimoto et al. 2008) and so on.

Literature on the subject can generally be separated between discrete and continuous settings. Discrete optimization models generally involve time-space networks that are less likely to present the timing issues studied in this article, whereas several continuous models have led to explicit timing problems with $P(t)$ features, as in Ichoua et al. (2003), Fleischmann et al. (2004), Donati et al. (2008) and Hashimoto et al. (2008). These models involve constraints of the type $t_i + p_i(t_i) \le t_{i+1}$ within a timing formulation with other additional features. Very often, the *FIFO* assumption on functions p_i is made:

$$\text{FIFO assumption: } \forall i \; x \ge y \Rightarrow x + p_i(x) \ge y + p_i(y) \tag{30}$$

FIFO implies that any delay in an activity execution date results in a delay in its completion date. Such assumption is meaningful in real life routing, as two vehicles that behave similarly on the same route are supposed to remain in the same arrival order, whatever congestion happens (Ichoua et al. 2003). However, there are situations where FIFO is not relevant, in transportation modes such as rail, where express trains can eventually arrive at destination before slower trains with earlier departures for the same destination.

Time-dependent processing time features are generally assumed to result in more complex timing problems. However, one should clearly identify the source of the difficulty, which is frequently imputable to the calculation and access to $p_{ij}(t)$ through the search, and not necessarily

to the timing problem solving method. Assuming that $p_{ij}(t)$ can be evaluated in constant time, and under FIFO, $\{D|R, P(t)\}$ is still solvable in $O(n)$ by means of a *minimum idle time policy* (Fleischmann et al. 2004). In the same spirit, Donati et al. (2008) apply the time-slack approach of Savelsbergh (1992) for $\{\emptyset|TW, P(t)\}$. However, dedicated methodologies are necessary for other settings such as $\{DUR|TW, P(t)\}$, and re-optimization procedures with $P(t)$ (Section 8).

A general time-dependent timing problem with costs related to service and departure dates is also tackled by Hashimoto et al. (2008). All functions considered are non-negative, piecewise linear, and lower semicontinuous. The authors propose a dynamic programming approach, which extends the method of Section 5.4. It involves the functions $F_i(t)$, which represent the minimum cost to proceed the subsequence of activities $(a_1 \dots a_i)$, while starting the last activity before t ($t_i \leq t$). These functions can be computed by means of the forward dynamic programming formulas of Equations (31-32), and the optimal solution cost is given by $F_n(+\infty)$.

$$F_1(t) = \min_{0 \leq x \leq t} \{c_1(x)\} \tag{31}$$

$$F_i(t) = \min_{0 \leq x \leq t} \{c_i(x) + \min_{x' + p_i(x') \leq x} F_{i-1}(x')\} \qquad 1 < i \leq n \tag{32}$$

It is remarkable that, under the assumption of Equation 33, which is slightly weaker than FIFO, the method can be implemented in $O(n(\varphi_c + \varphi_p))$, where φ_c and φ_p denote the total number of pieces in cost and processing-time functions. This method for $\{\Sigma c_i(t_i)|P(t)\}$ thus present the same quadratic complexity as in the case without time-dependency (Section 5.4).

$$\text{(HYI) assumption: } \forall i\ x + p_i(x) = y + p_i(y) \Rightarrow x + p_i(x) = z + p_i(z)\ \forall z \in [x, y] \tag{33}$$

However, when the previous assumption does not stand, the dynamic programming method of Hashimoto et al. (2008) is not polynomial, and the question remains open whether $\{\Sigma c_i(t_i)|P(t)\}$ is polynomially solvable or not in this case. Hypotheses on processing times functions, such as FIFO, are then crucial to guarantee efficient solution procedures.

6.5 Time-lags

The two-dimension features considered until now involved linking constraints and objectives between the first and last task, in the case of DUR, or between pairs of successive variables in the case of NWT and IDL. We now address the time lags TL feature, which involves any difference of two execution dates in constraints or objectives. This feature is thus naturally a generalization of NWT, IDL, DUR and processing-times.

To the best of our knowledge, the first research on time lags has been conducted by Mitten (1959) for flowshop scheduling problems. This feature has been used since to model many problem characteristics in various domains, such as the deterioration of food or chemical products, glue drying, customer requirements in some dial-a-ride problems, elevator dispatching, quarantine durations and so on. Time-lag scheduling problems on a single machine have also been shown by Brucker et al. (1999) to generalize all shop, multi-purpose machines, and multi-processor tasks scheduling problems. Hence, the resulting timing problems with TL are also likely to be difficult.

The most basic problem with TL feature relates to feasibility checking under time-lag constraints of the form $t_i + \delta_{ij} \leq t_j$. When $\delta_{ij} \geq 0$, the constraint is called positive time-lag, and corresponds to a minimum delay between activities a_i and a_j, whereas $\delta_{ij} \leq 0$ corresponds to a negative time-lag, and involves a maximum delay of $-\delta_{ij}$ between activities a_j and a_i. Equality

constraints $t_i + \delta_{ij} = t_j$ involve both positive and negative time-lags. The resulting timing problem $\{\emptyset|TL\}$ can be seen as a special case of project scheduling on a chain of activities, and the *METRA potential method (MPM)* of Roy (1959, 1962) can be applied. Following MPM, time lag constraints can be represented on a graph $G = (V, A)$ where each activity a_i is associated with a node $v_i \in V$, and each arc (v_i, v_j), associated with a weight w_{ij}, represents a temporal constraint of the form $t_j - t_i \geq w_{ij}$. Feasibility of the timing problem $\{\emptyset|TL\}$ is then equivalent to the non-existence of a positive length cycle in this graph (Bartusch et al. 1988, Dechter et al. 1991). The algorithm of Floyd-Warshall can be employed to solve this problem in $O(n^3)$, but the longest path procedure of Hurink and Keuchel (2001), also in $O(n^3)$, is shown to provide faster results in practice. Potts and Whitehead (2007) also consider a coupled-operation scheduling problem with only $n/2$ time-lag constraints, and timing feasibility is checked in $O(n^2)$. The authors underline the computational burden of such timing algorithms, which strongly degrades the performance of neighborhood searches or branch and bound procedures.

In the context of dial-a-ride problems, Hunsaker and Savelsbergh (2002) also study a case of $\{\emptyset|TL, TW\}$ timing. Activities represent customer requests on drop-on and drop-off services, which occur by pairs, such that any drop-on activity always precedes its corresponding drop-off activity in the sequence. Each such pair of activities is linked by a single positive time-lag constraint. The total number of time-lag constraints is thus $n/2$. The problem also involves time-windows and maximum waiting times for each activity. The authors claim that the resulting timing feasibility problem can be solved in three passes on the sequence of activities with linear complexity. Yet, the algorithm presents a small flaw which is straightforward to correct (Tang et al. 2010), but results in a complexity of $O(n^2)$. An alternative algorithm, proposed by Haugland and Ho (2010), improves this complexity to $O(n \log n)$ by means of heap data structures.

Finally, Cordeau and Laporte (2003) and Berbeglia et al. (2010) consider a dial-a-ride setting with an additional duration constraint on the whole vehicle trip duration. The authors consider a relaxed problem through the search, with a hierarchical objective. Total trip duration infeasibility is minimized, then time-windows infeasibility and finally time-lag infeasibility, that is the timing problem $\{DUR > D > TL|R\}$. The algorithm first minimizes duration and time-window infeasibility as in Section 6.1, then iteratively delays some drop-on services to reduce time-lag infeasibility without increasing any other violation. A computational complexity of $O(n^2)$ is achieved. It was observed (Private communication 2010), however, that the previous approach only guarantees optimality under an additional assumption that we call *LIFO*, and which requires that for any $1 \leq i < j < k < l \leq n$, no activities a_i, a_j, a_k, a_l present "entangled" time-lag constraints of the form $t_k - t_i \leq \delta_{ik}$ and $t_l - t_j \leq \delta_{jl}$. An instance presenting this property is illustrated in the upper part of Figure 7, while at the bottom is represented an arbitrary instance presenting entangled constraints.

The LIFO assumption is frequently respected in the vehicle routing literature, especially in pickup and deliveries services, or in presence of complex loading constraints. In this scope, the last object or customer received in the vehicle is the first one to leave. Without this assumption, the difficulty of many problems with time-lags strongly increases, and no specialized efficient algorithm is actually known for $\{DUR > D > TL|R\}$ and similar problems.

6.6 Separable costs by pairs of variables

Separable costs by pairs of variables $\sum c_{ij}(t_i, t_j)$ generalize all problems combining single or two-dimension features. The timing problem with this feature alone is NP-hard, as it includes

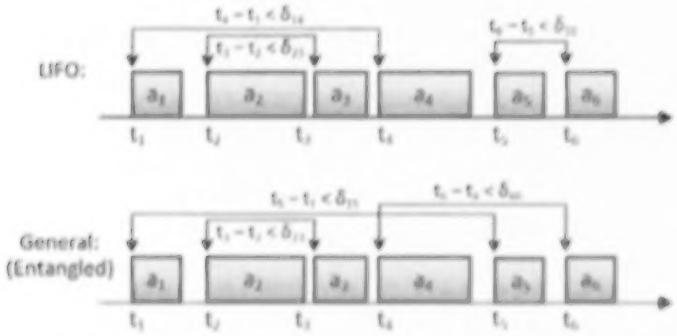


Figure 7: Timing problems with time lags: illustration of the LIFO property

$\{\Sigma c_i(t_i), \Sigma c_i(\Delta t_i)|\emptyset\}$ as special case (see Section 6.2). Under convexity or linearity assumptions on the objective function, the timing problem $\{\Sigma c_{ij}^{CVX}(t_j - t_i), \Sigma c_i^{CVX}(t_i)|\emptyset\}$ given in Equation (34), is equivalent to the *convex cost dual network flow problem*, for which weakly polynomial algorithms are available (Ahuja et al. 2003). This problem constitutes one of the most general timing setting to remain polynomially solvable.

$$\min_{t=(t_1 \ldots t_n)} \quad \sum_{(i,j)\in Q} c_{ij}^{CVX}(t_j - t_i) + \sum_{1\leq i\leq n} c_i^{CVX}(t_i) \tag{34}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{35}$$

7 Conclusion on "stand-alone" timing methods

In contrast to single-dimension features, which appeared as fairly well addressed in Section 5 by means of a few algorithms and concepts, two-dimension features lead to more diverse problem structures and algorithms. Simple cases with duration minimization or time-dependent processing times can eventually be solved in linear time, but other problems with time-lags features actually require $O(n^3)$ algorithms to be solved exactly. Although polynomial, the latter methods can be impracticable in the context of local searches or branch-and-bound approaches.

Many practical timing settings result in models with linear constraints, and linear or separable convex objectives. For these problems, the linear and convex programming theory ensures weakly polynomial resolvability, and provides general solving methods (Khachiyan 1979, Karmarkar 1984, Hochbaum and Shanthikumar 1990). Some of the most complex timing settings, however, such as $\{\Sigma c_i(t_i), \Sigma c_i(\Delta t_i)|\emptyset\}$, are NP-hard, while for other problems, such as $\{\Sigma c_i(t_i)|P(t)\}$ with general piecewise linear functions $P(t)$, the existence or non-existence of polynomial algorithm is still open. Timing settings thus lead to a rich variety of problem structures and complexities.

In all these cases, whether polynomial algorithms are available or not, research is still open to provide more efficient algorithms exploiting the particular structure of the features and problems at hand. The present paper has already contributed by building a formalism, a classification of features, timing problems and methods. We gathered the most efficient timing approaches from numerous previously separated fields of research, to tackle both specialized timing settings, and more general features. Now research can turn on filling the gaps which

have been emphasized in this review, and which continue to appear, following the rich variety of actual application cases with time constraints emerging nowadays.

Last but not least, we mention finally a key direction of research, which involves more efficient timing solving, not from a stand-alone view, but considering the general context of a local search or branch-and-bound approaches. In this scope, closely related timing problems are repetitively solved for each move in the neighborhood, or at each node of the search tree. We dedicate a last Section of this paper to efficient re-optimization methods for these important settings.

8 Timing re-optimization

In the first part of this article, we examined how to address timing problems as a stand-alone issue. Yet, most neighborhood-search-based heuristics, metaheuristics, and some exact methods require to solve iteratively a large number of timing instances in close relationships. In this case, solving each timing problem "from scratch", without exploiting any knowledge on previous resolution, can result in a huge loss of information and in redundant computations.

Most local searches for routing and scheduling problems (see the book of Hoos and Stützle 2005 for a thorough presentation of LS) indeed rely on a neighborhood based on a limited number of sequence changes. One or several timing subproblems are solved for each neighbor solution, in order to determine the cost or the feasibility of the sequences it involves. Two classical neighborhoods, dedicated respectively to exchange the tails of two vehicle routes in VRP context (2-opt* of Potvin and Rousseau 1995), and relocate a sequence of activities (Or-exchange of Or 1976), are illustrated in Figure 8. It is noticeable that large subsequence of activities (Seq.A ... Seq.D on the figure) are shared by the successive timing subproblems to solve.

Figure 8: Sequence invariants in 2-opt* and relocate moves

Also, for problems presenting sequencing issues, several branch and bound based searches involve branching decisions on sequences characteristics, such as precedence relationships, or arc or path setting. Timing subproblems can then arise at nodes of the search tree, when evaluating the cost or feasibility of sequences produced during the search, for lower bound computation, branch pruning (Hoogeveen and Van De Velde 1996, Sourd and Kedad-Sidhoum 2003), as well as in column generation approaches (Desrochers et al. 1992, Prescott-Gagnon et al. 2009, Baldacci et al. 2011). In the latter cases, the search for improving columns involves elementary shortest path problems with combined timing decisions and resource constraints.

In the previous cases, it is noticeable that numerous closely related timing problems must be solved: large subsequences of consecutive activities remain unchanged, and only a minor proportion of problem parameters (reduced cost values for column generation) is impacted. In

order to solve more efficiently these series of similar timing problems, several authors propose to keep *meaningful data* on the global search process to save computations. Actually, neighborhood searches have largely benefited from these techniques, as move evaluation (and thus timing solving) is responsible for the largest part of the computational effort. Therefore these *re-optimization* methodologies can lead to dramatic reductions in the computational burden of algorithms.

We now formally define these serial timing problems in Section 8.1, and present a general framework for re-optimization methods based on sequence concatenations in Sections 8.2-8.3. Links with related re-optimization methodologies are analyzed in Section 8.4, before reviewing or introducing efficient concatenation-based re-optimization methods for each major timing feature and related problems in Section 8.5-8.6. The set of timing re-optimization methods introduced in the remaining sections are fundamental to design efficient neighborhood searches for a large range of vehicle routing and scheduling problems.

8.1 Problem statement: Serial timing

This section formally defines serial timing problems, and presents the general concepts of re-optimization from the literature.

Definition 5 (Serial timing). *Let $\mathcal{T}$ be an* incumbent timing problem *with n activities $(a_1 \dots a_n)$, sequence-dependent processing-times p_{ij}, and additional features with their data, characteristic functions $f_i^x(\mathbf{t})$, and role involving new constraints $f_y^x(\mathbf{t}) \leq 0$ for $y \in [\![1, m_x]\!]$ or contributing to the objective with a value value $\bar{f}^x(\mathbf{t})$ (see Section 2). N permutation functions $\sigma^k : \{1 \dots n\} \to \{1 \dots n\}$ for $k \in [\![1, N]\!]$, are also given. The* serial timing *problem then involves solving the timing subproblems $\mathcal{T}^k$ of Equations (36-38), for $k \in [\![1, N]\!]$.*

$$(\mathcal{T}^{\mathbf{t}}): \quad \min_{\mathbf{t}=(t_1,\dots,t_n)} \quad \sum_{f^x \in \mathcal{F}^{\text{OBJ}}} \alpha_x \bar{f}^x(\mathbf{t}) \tag{36}$$

$$s.t. \quad t_{\sigma^k(i)} + p_{\sigma^k(i)\sigma^k(i+1)} \leq t_{\sigma^k(i+1)} \qquad 1 \leq i < n \tag{37}$$

$$f_y^x(\mathbf{t}) \leq 0 \qquad f^x \in \mathcal{F}^{\text{CONS}}, \ 1 \leq y \leq m_x \tag{38}$$

Sequence-dependent processing times are considered here, in relation to a large range of applications in the field of vehicle routing, which heavily rely on re-optimization methods, and for which this characteristic is central. Sequence-dependency affects the methods at play only in some timing settings, which are explicitly mentioned in this paper.

Several types of re-optimization approaches have been developed in the literature to take advantage of the information developed during the successive subproblem solving. To solve similar timing instances, a first approach involves to re-arrange previously developed schedules in relation to the new settings. Several timing features, especially, lead to network flows or shortest-path formulations, on which re-optimization methods related to a change of arcs or costs in the network have been thoroughly studied (Goto and Sangiovanni-Vincentelli 1978, Pallottino 2003, Frangioni and Manca 2006, Miller-Hooks and Yang 2005). Most timing problems can also be modeled as linear programming models, on which sensitivity analysis and warm start following a problem modification has been studied for long, and can eventually be tackled by means of a primal-dual simplex algorithm. Finally, a last methodology, on which we focus in the following, is based on the simple observation that a permutation of activities can be assimilated to a concatenation of some subsequences of consecutive activities. Hence, keeping

information on subsequences (e.g., dynamic programming computations), can lead to significant speed up in solving (Kindervater and Savelsbergh 1997a, Cordone 2000). We introduce in the next sections a framework for these concatenation properties and the re-optimization approaches which follow.

8.2 Breakpoints and concatenations

We first introduce some vocabulary, and then emphasize the links between operations on sequences of activities, such as activity relocations, or changes of order relations, and properties of the resulting permutation functions. These observations lead to efficient re-optimization approaches "by concatenation".

Definition 6 (Permutation breakpoints). *Let $\sigma : \{1 \dots n\} \to \{1 \dots n\}$ be a permutation. Any integer b such that $\sigma(b) + 1 \neq \sigma(b+1)$ and $1 \leq i < n$ is called a breakpoint of σ, and corresponds to non consecutive values in the permutation representation. The total number of breakpoints of σ is notated $b(\sigma)$, and we denote them as $b_1^\sigma \dots b_{b(\sigma)}^\sigma$ in increasing order.*

For example, the permutation $\sigma_0 : \{1,2,3,4,5,6\} \to \{4,\mathbf{5},\mathbf{3},1,\mathbf{2},6\}$ has three breakpoints (indicated in boldface): $b_1^{\sigma_0} = 2$, $b_2^{\sigma_0} = 3$ and $b_3^{\sigma_0} = 5$. We now show the links between classical operations on activity sequences and the resulting permutation function properties in terms of breakpoints:

Lemma 1 (Order changes). *Let A' be an activity sequence obtained from A by changing l order relations, and $\sigma_{A \to A'}$ the permutation function at play, then $b(\sigma_{A \to A'}) = l$.*

Lemma 2 (Activity relocations). *Let A' be an activity sequence obtained from A by relocating l activities, and $\sigma_{A \to A'}$ the permutation function at play, then $b(\sigma_{A \to A'}) \leq 3l$.*

Indeed, any change in order relation results in exactly one breakpoint, while any relocation of activity can be assimilated to at most three changes of order relations, and thus results in three breakpoints. Situations where k order relations or activities are changed from one timing problem T to another problem T' frequently occur in the context of neighborhood search methods working on sequences of activities, where timing subproblems must be solved to evaluate cost or feasibility of each sequence explored in the neighborhood. The interest of breakpoints lies in the following proposition which, although straightforward, provides the basis of re-optimization methods working by concatenation:

Proposition 1. *Let $\sigma : \{1 \dots n\} \to \{1 \dots n\}$ be a permutation with breakpoints $b_1^\sigma \dots b_{b(\sigma)}^\sigma$. Let A be a sequence of n activities, then $A' = \sigma(A)$ corresponds to the concatenation of exactly $b(\sigma)+1$ subsequences of consecutive activities in A, as presented in Equation 39 (a dummy breakpoint $b_0^\sigma = 1$ represents the beginning of the sequence).*

$$A' = \bigoplus_{l=0 \dots b(\sigma)-1} (a_{\sigma(b_l^\sigma + 1)} \cdots a_{\sigma(b_{l+1}^\sigma)}) \tag{39}$$

Any bounded number of operations transforming an activity sequence A into A' (relocation of activities, or changes of order relations) thus involves a permutation function with a bounded number of breakpoints, such that A' can be seen as concatenation of a bounded number of subsequences of A. Pre-processed informations from a bounded number of subsequences can then be exploited to produce information on their concatenation, to solve the timing subproblems without browsing all activities of the sequence at play. The next section formalizes the re-optimization operations and algorithms which can be developed to this extent.

8.3 Re-optimization "by concatenation"

Re-optimization by concatenation can be formalized around a set of four basic re-optimization operators, given in Table 2, which are used for data acquisition and exploitation on subsequences of consecutive activities. As the goal of this methodology is to avoid redundant computations, it is particularly crucial for all these operators to maintain the integrity of the input data.

Table 2: Re-optimization operators

Initialization:	Initialize the data $\mathcal{D}(A)$ of a sequence containing a single activity.
Forward extension:	Given the data of a sequence $A = (a_i \dots a_j)$, and an activity a_k to be added forward, determine the data $\mathcal{D}(A')$ for the resulting sequence $A' = (a_k, a_i \dots a_j)$
Backward extension:	Given the data of a sequence $A = (a_i \dots a_j)$, and an activity a_k to be added backwards, determine the data $\mathcal{D}(A')$ for the sequence $A' = (a_i \dots a_j, a_k)$.
Evaluate concatenation:	From the data $\mathcal{D}(A_l), l \in 1...L$ of one, two, or several activity sequences, evaluate the optimal cost (and eventually the optimal solution) of the timing problem involving the concatenation of these sequences

It should first be noted that the *forward extension* (respectively *backward extension*) operation of Table 2 directly derives from forward or backward dynamic programming concepts. Bi-directional dynamic programming approaches can also be assimilated to both forward and backward extension operations, as well as a single concatenation evaluation to provide the optimal solution (Righini and Salani 2006). The re-optimization approach "by concatenation", illustrated in Algorithm 1, is also based on these operators. Data is built on subsequences of the incumbent timing problem $\mathcal{T}$, by means of the forward and backward extension operators. These data, developed during a preprocessing phase or eventually through the search, are then used to solve repetitively all the derived subproblems using the *evaluate concatenation* operator.

Algorithm 1 Re-optimization

1: Build re-optimization data on subsequences of the *incumbent timing problem* $\mathcal{T}$, using *initialize*, and *forward extension* or *backward extension*.
2: For each timing subproblem $\mathcal{T}^k$, $k \in ||1, N||$,
3: Determine the breakpoints involved in the permutation function σ^k
4: Evaluate the optimal cost of $\mathcal{T}^k$, as the concatenation of $b(\sigma) + 1$ activity subsequences from $\mathcal{T}$ (see Equation 39), relying on the subsequences re-optimization data.

The efficient applicability of Algorithm 1 directly relies on the potential to develop concatenation operations which are less computationally complex than stand-alone methods. One should also pay attention to the price to pay in terms of data computing on subsequences, and whether the resulting computational effort is dominated by the quantity of derived timing subproblems to solve. There is no unique way to apply this method: problem specific design choices arise for example when determining the nature of the data, the subsequences on which it is computed, as well as the instants dedicated to data computation.

We therefore illustrate an application in neighborhood search for routing problems with time constraints on routes. In this scope, a local search improvement procedure based on sequences changes leads to a number of timing subproblems proportional to the number of neighborhood solutions to explore. In the VRP or scheduling literature, the number of derived

timing subproblems is often $N = \Omega(n^2)$, and the derived activity sequences do not involve the concatenation of more than $k = 2, 3$ or 4 subsequences of the original problem.

Attempting to solve each timing subproblem independently, the overall complexity of a neighborhood exploration is $Nc(T)$, $c(T)$ being the computational complexity of one stand-alone timing solution procedure. A straightforward re-optimization approach consists in exhaustively computing the data for each of the $n(n-1)$ subsequences of consecutive activities from $\mathcal{T}$, and then use it to evaluate all moves. Data computation is straightforward to perform in $O(n^2c(I) + n^2c(F/B))$, $c(I)$, $c(F/B)$ being respectively the computational complexity of initialization and either forward or backward extension. The overall complexity of the new neighborhood exploration procedure is thus $O(n^2c(I) + n^2c(F/B) + Nc(EC))$, $c(EC)$ being the complexity for evaluating concatenation of less than 4 subsequences. Assuming that $N = \Omega(n^2)$, the computational complexity of neighborhood evaluation becomes $O(N[c(I) + c(F/B) + c(EC)])$ for re-optimization methods instead of $Nc(T)$ for independent solving. Re-optimization operators being less computationally complex than stand-alone methods, the resulting approach is likely to lead to reduced computational effort.

For some settings, such as $\{\emptyset|MTW\}$ and most problems with two-dimension features, concatenation operators involving more than two sequences are not actually available or not computationally suitable for efficient re-optimization. In such cases, forward and backward propagation can be used to a larger extent, along with concatenations of only two subsequences, to perform the timing subproblem evaluations. Such an example is given by the lexicographic search of Savelsbergh (1985), which enables to evaluate timing subproblems associated to well known neighborhoods for vehicle routing problems exclusively by means of concatenation of two subsequences. Designing efficient evaluation orders in such contexts becomes a problem dependent issue, clearly impacted by the complexity of the operators at hand, and the nature of the permutations involved.

Finally, if concatenation of many subsequences can be operated efficiently, but that data creation constitutes the bottleneck in terms of computational effort, the subset of subsequences involved can be restricted to $O(n^{4/3})$ or $O(n^{8/7})$, using the hierarchical approach of Irnich (2008a). The relevant data to compute can also be tailored relatively to the neighborhoods at play. For example, the 2-opt* move presented in Figure 8 involves only the concatenation of subsequences containing the first or last activity, that we call *prefix* or *suffix subsequences*. The number of such subsequences requiring data computation is thus reduced to $O(n)$.

8.4 Related literature

A large range of problems in routing and scheduling with idle-time are tackled using local search on sequences. Serial timing issues thus frequently arise in these fields.

Generalizing the seminal work of Savelsbergh (1985, 1992) on efficient time-window feasibility checking and duration minimization in edge exchange based local searches for vehicle routing, Kindervater and Savelsbergh (1997a) proposed a framework to manage several constraints on vehicle routes, such as precedence constraints, time-windows, collection and deliveries under capacity constraints. Most of these constraints either correspond explicitly to timing features (i.e., time-windows) or result in models tackled in this paper. To perform efficient feasibility checking, the authors develop *global variables* on partial routes, which are used in concatenation operations to evaluate moves consisting of a constant number of edge exchanges. Move evaluations are performed in lexicographic order, to allow calculation of the global variables through the search.

Simultaneously, Cordone (2000) reports in a technical note similar concepts of data computation and concatenation operators. Although the proposed methodology is not extended to a large variety of constraints as in Kindervater and Savelsbergh (1997a), it explores different possibilities related to the concept of macro-nodes: if data on subsequences have the same structure as data on activities, subsequences of activities can be eventually replaced by equivalent single activities, called macro-nodes, in the method. Concatenation concepts in this case enable to temporarily reduce the problem size by collapsing nodes into macro-nodes, opening the way to algorithms based on aggregation of activities and multi-level approaches (Bean et al. 1987, Walshaw 2002, Elhallaoui et al. 2005).

Campbell and Savelsbergh (2004) present a compilation of efficient insertion heuristics for many vehicle routing problems with additional characteristics such as shift time limits, variable delivery quantities, fixed and variable delivery times, and multiple routes per vehicle. These methods iteratively create solutions by adding customers to the routes. The authors show that by managing global data on the routes, the cost of feasibility of customer insertions can be evaluated in amortized $O(1)$ for many of these settings. This approach can be seen as a specialization of the evaluate concatenation operator applied to a couple of partial routes with an intermediate activity.

A rich body of dynamic-programming based timing algorithms is also presented in Ibaraki et al. (2005), Hashimoto et al. (2006, 2008), Ibaraki et al. (2008) and Hashimoto et al. (2010). Forward and backward propagation is used, with an additional operator (named *connect* in Hashimoto 2008) to manage concatenation of two subsequences, thus leading to efficient re-optimization approaches by concatenation for several timing problems involving piecewise linear functions.

Finally, the resource extension framework of Desaulniers et al. (1998) models many constraints and objectives on sequences of activities as resources that are subject to window constraints, and extended from one activity to the next by means of resource extension functions (REFs). This framework proved extremely efficient to model many crew scheduling and routing problems, and solve them by column generation (Desaulniers et al. 2005). It has also been recently extended by Irnich (2008a,b) for efficient neighborhood search design under various constraints on routes, such as load dependent costs, simultaneous pickup and deliveries, maximum waiting times and times on duty. To that extent, REFs "generalized to segments" are built on subsequences to characterize the resource consumption to the last activity of a subsequence given the resource consumption to the first activity. Inverse resource extension functions are also introduced, providing upper bounds on feasible resource consumption to the first activity from the resource availability to the last activity. Developing this data on subsequences enables then to evaluate efficiently the cost or feasibility of local search moves. This framework, however, requires rather restrictive conditions: the existence of REFs which can be generalized to subsequences and inversed, and such that the resource extensions functions on subsequences take the same form as the resource extensions for a single activity. These conditions are satisfied only by a limited subset of the timing problems and features introduced previously, such as $\{\emptyset|TW\}$, $\{\emptyset|MTW\}$ or $\{DUR|TW\}$.

Several general methodologies thus exist in the literature to tackle timing problems within a neighborhood search context. However, these approaches were restricted either by the types of concatenations allowed, the assumptions made on features, or the applicability of the models to a wide range of constraints. Less specialized, the timing formalism we propose, and its generalization to re-optimization operators by concatenation in the spirit of Kindervater and

Savelsbergh (1997b), is developed for a wider range of timing settings. As shown in the next Sections (8.5-8.6) through the study of state-of-the-art re-optimization methodologies with various features, the proposed framework unifies previous successful concepts, and provides a line of thought for the development of efficient algorithms for various timing problems. It can also be seen as a generalization of Irnich (2008a,b) concepts, applied to timing problems. Indeed, generalized resource extensions functions and their inverse provide, when they exist, the suitable data for re-optimization. We now review, analyze and develop re-optimization approaches for main timing features and problems, describing the data and operators involved, and their performance. As in the first part of this paper, this analysis is organized by increasing order of complexity and feature dimension.

8.5 Re-optimization for single-dimension features

Constant activity costs & cumulative resources. In presence of constant activity costs (or in a more general setting any cumulated resource such as distance or load under a global constraint), evaluating the total cost of a solution "from scratch" would involve to browse each activity and cumulate the costs (or the resource), resulting in a linear complexity. A simple and efficient re-optimization approach involves to manage the following data on subsequences:

Data and computation: Cost $C(A_i)$ (or total resource used) for each subsequence A_i.

Evaluate concatenation: Concatenating k subsequences involves k sums: $C(A_1 \oplus \cdots \oplus A_k) = \sum C(A_i)$. Evaluating the concatenation of a bounded number of subsequences can thus be performed in a bounded number of operations, leading to $O(1)$ complexity for move evaluation when the data is available. Data can be processed in amortized constant time for each move during a local search procedure for many classic neighborhoods, using a *lexicographic order* (Kindervater and Savelsbergh 1997a) for move evaluation, or developed in a preprocessing phase for a complexity of $O(n^2)$, which is often dominated by the neighborhood size.

Weighted execution dates & non-decreasing linear costs. Timing subproblems with non-decreasing linear costs $c_i(t_i) = w_i t_i + c_i$ with $w_i \geq 0$ for all i can be solved efficiently by means of the following re-optimization operators.

Data: Total processing time $T(A_i)$ for all the activities of the sequence A_i but the last one. Waiting cost $W(A_i)$ related to a delay of one time unit in the sequence processing, and sequence cost $C(A_i)$ when started at time 0.

Data computation and evaluate concatenation: For a sequence A with a single activity, $T(A) = 0$, $W(A) = w_{A(1)}$ and $C(A) = c_{A(1)}$. Equations (40-42) enable both to evaluate the cost of concatenation and compute the data for sequences with more activities.

$$W(A_1 \oplus A_2) = W(A_1) + W(A_2) \tag{40}$$

$$C(A_1 \oplus A_2) = C(A_1) + W(A_2)(T(A_1) + p_{A_1(|A_1|)A_2(1)}) \tag{41}$$

$$T(A_1 \oplus A_2) = T(A_1) + p_{A_1(|A_1|)A_2(1)} + T(A_2) \tag{42}$$

The previous equations have been formulated to also manage sequence dependent processing times. It is remarkable that, in this case, the re-optimization data simply consists of a generalization of single activity characteristics to sequences of activities. It is also expressed as a finite number of coefficients, and allows for $O(1)$ concatenation operations. In the framework of Irnich (2008b), this *generalization to segments* gives the possibility to "aggregate" some sequences of

nodes to consider those as single activities during the search, thus temporarily reducing the problem size.

Time-window feasibility check. Savelsbergh (1985) opened the way to efficient feasibility checking with regards to time-windows in the context of local search. In the subsequent work of Kindervater and Savelsbergh (1997a), the following re-optimization data and operators are introduced.

Data: Total processing time $T(A_i)$ for all the activities of the sequence A_i but the last one. Earliest possible execution date $E(A_i)$ of the last activity in any feasible schedule for A_i. Latest possible execution date $L(A_i)$ of the first activity in any feasible schedule for A_i. A record $isFeas(A_i)$ valuated to true if and only if a feasible schedule for A_i exists.

Data computation and evaluate concatenation: For a sequence A with a single activity, $T(A) = 0$, $E(A) = e_{A(1)}$, $L(A) = l_{A(1)}$ and $isFeas(A) = true$. Equations (43-46) enable then to evaluate the cost of concatenation and compute the data for sequences with more activities in $O(1)$.

$$T(A_1 \oplus A_2) = T(A_1) + p_{A_1(|A_1|)A_2(1)} + T(A_2) \tag{43}$$

$$E(A_1 \oplus A_2) = \max\{E(A_1) + p_{A_1(|A_1|)A_2(1)} + T(A_2), E(A_2)\} \tag{44}$$

$$L(A_1 \oplus A_2) = \min\{L(A_1), L(A_2) - p_{A_1(|A_1|)A_2(1)} - T(A_1)\} \tag{45}$$

$$isFeas(A_1 \oplus A_2) \equiv isFeas(A_1) \wedge isFeas(A_2) \wedge E(A_1) + p_{A_1(|A_1|)A_2(1)} \leq L(A_2) \tag{46}$$

Earliness, tardiness, soft time-windows, and separable convex costs. Timing problems with tardiness $\{D|\emptyset\}$, earliness and tardiness $\{R, D(r_i = d_i)|\emptyset\}$, or with soft time-windows $\{TW|\emptyset\}$ in sequencing or vehicle routing problems have been often solved in a stand-alone way, using respectively an minimum idle time policy in $O(n)$, or variants of the PAV algorithm (Section 5.3) in $O(n \log n)$. Ibaraki et al. (2008) recently considered the efficient solving of the related serial timing problems, and attained a $O(\log n)$ amortized complexity per timing subproblem for rather general cases. This approach is applicable to $\{\Sigma c^{CVX}(t)|\emptyset\}$ with piecewise linear, non-negative, lower semicontinuous, convex activity costs. Another approach by Ergun and Orlin (2006) and Kedad-Sidhoum and Sourd (2010), presented later in this section, allows a calculation in $O(1)$ for some particular cases of lateness or (E/T) serial timing problems without sequence-dependency, and with specific permutation functions.

Ibaraki et al. (2008) develop data on subsequences as piecewise linear convex functions. The functions involved are exactly the ones previously described in Section 5.4, and are represented and stored by means of binary search trees. In order to reduce the preprocessing effort, they are computed only on prefix or suffix subsequences, that contains respectively the first or the last activity of the incumbent timing problem.

Data: Total duty time $T(A_i)$ on a partial sequence A_i. Optimal cost $F(A_i)(t)$ of a schedule for A_i, when the first activity is executed before $t(t_{A_i(1)} \leq t)$. Optimal cost $B(A_i)(t)$ of a schedule for A_i, when the last activity is executed after t.

Data computation: Data computation of $F(A_i)(t)$ and $B(A_i)(t)$ is respectively performed by means of forward and backward dynamic programming (Equations 19-20). Using efficient structures as in Ibaraki et al. (2008) to represent piecewise linear functions, data computation can be performed in $O(\varphi_c \log \varphi_c)$ for a subsequence, where φ_c is the total number of pieces in the cost functions of the timing subproblem.

Evaluate concatenation: Equation (47) returns the optimal cost $Z^*(A_1 \oplus A_2)$ of the timing problem related to the concatenation of two sequences. With the same notations as previously, an amortized complexity of $O(\log \varphi_c)$ is attained. When the number of pieces of cost functions is linear in the number of activities, as in earliness, tardiness or soft time-window timing settings, a complexity of $O(\log n)$ is attained. Evaluations of concatenations involving three or four subsequences can be performed with the same efficiency. We refer to Ibaraki et al. (2008) for details on the equations required.

$$Z^*(A_1 \oplus A_2) = \min_{t \geq 0} \{F(A_1)(t) + B(A_2)(t) + p_{A_1(|A_1|)A_2(1)})\} \tag{47}$$

The approach of Ibaraki et al. (2008) is fairly general, enabling to tackle earliness, tardiness, and soft time-windows features for any combination of subsequence concatenation. Yet, the resulting complexity in higher than in the case of other timing problems, like hard time-window checking, which led to re-optimization operators in $O(1)$. This logarithmic complexity is related to the calls to convex functions, stored in trees structures, that constitute the subsequence data.

Only Ergun and Orlin (2006) and Kedad-Sidhoum and Sourd (2010) have actually reported methods with an "evaluate concatenation" operator working in amortized constant time for $\{D|\emptyset\}$ and $\{D, R(d_i = r_i)|NWT\}$ serial timing respectively, and under particular types of permutation functions (corresponding to neighborhood searches based on swap, insert as well as compound moves). The cornerstone of these two approaches is that they call the functions $B(A_2)(t)$ associated to subsequences by series of $O(n)$ increasing values of t, thus enabling to call the piecewise convex functions in amortized constant time. The methodology requires some orderings, which are performed in a pre-processing phase in $O(n \log n)$. Such complexity is generally dominated by the number N of timing subproblems. It is actually an open question whereas this type of approach can be extended to more general problems. This approach is likely to become far more complex when tackling sequence dependent processing times in particular. Approximate serial timing procedures in $O(1)$ can also be used when computational time is critical (Taillard et al. 1997).

Separable cost functions (including multiple time-windows). Ibaraki et al. (2005) and Hendel and Sourd (2006) introduced simultaneously, respectively in the domains of VRP and scheduling, a re-optimization approach in $O(\varphi_c)$ for timing problems with piecewise linear cost functions. This re-optimization approach involves the same quantities $T(A_i)$, $F(A_i)(t)$ and $B(A_i)(t)$ as in the piecewise linear case presented in the previous sub-section. In this case, simple linked lists are sufficient for function representation. Evaluating the concatenations is also performed by means of Equation (47), resulting in $O(\varphi_c)$ complexity.

Perspectives: re-optimization for single-dimension features. For most single-dimension features, re-optimization approaches lead to large computational complexity reductions, frequently by a factor of n or φ_c. There are still some particular cases, such as the timing problem $\{D|R\}$, which can be solved "from scratch" in $O(n)$, but only allows a $O(\log n)$ re-optimization methodology for general settings. This timing problem is involved in many VRP heuristics, which temporarily relax time-window constraints to allow lateness in the course of the search, thus increasing the connectivity of the search space. Yet, this choice of relaxation results in an increased complexity of $O(\log n)$, against $O(1)$ for simple feasibility checking, or for a different relaxation scheme (see Section 8.6). Choices of relaxation at the heuristic design level must

thus be taken carefully. Also, further work on $\{D|R\}$, to either show the impossibility to reach $O(1)$ re-optimization methodologies in the general case, or to provide new methods, would be extremely valuable.

8.6 Re-optimization for two-dimension features

Total duration and idle time. Savelsbergh (1992) enabled to perform move evaluations in $O(1)$ for VRP's involving $\{\emptyset|DUR, TW\}$ or $\{DUR|TW\}$ as subproblems. Noteworthy is the approach of Kindervater and Savelsbergh (1997a), which derives from this work, and has been presented in Section 8.5 for $\{\emptyset|TW\}$. Indeed, the total sequence duration can be retrieved in $O(1)$ from the re-optimization data of Equations (43-46) using Equation (48).

$$DUR(A_i) = \max\{E(A_i) - L(A_i), T(A_i)\} \tag{48}$$

Flexible processing times. The flexible processing time feature involves separable functions of successive activity execution dates. Complex problems are raised when this feature is coupled with time-window constraints as in $\{\Sigma c_i(\Delta t_i)|TW\}$ (Equations 49-50), or with separable activity execution costs. The total order constraints can be directly taken into account in the objective, with cost functions c_{ij} such that for $\Delta t < 0$, $c_{ij}(\Delta t) = +\infty$.

$$\min_{t_1 \dots t_n} \quad \sum_{i=1}^{n} c'_{\sigma_i \sigma_{i+1}}(t_{\sigma_{i+1}} - t_{\sigma_i}) \tag{49}$$

$$s.t. \quad e_i \leq t_i \leq l_i \qquad\qquad 1 \leq i \leq n \tag{50}$$

We first present a re-optimization approach for a particular shape of cost functions c_{ij}, introduced by Nagata (2007) in the context of VRP with time-window constraints, as an alternative relaxation of time-windows to be used during heuristic search. Instead of allowing earliness or lateness with respect to time-window constraints, penalized processing times reductions are allowed. The resulting cost functions are given in Equation 51.

$$c'_{ij}(\Delta t) = \begin{cases} 0 & \text{if } \Delta t \geq p_{ij} \\ \alpha(p_{ij} - \Delta t) & \text{otherwise} \end{cases} \tag{51}$$

This case is relatively exotic in the timing context, as no limit is fixed on the amount of processing time reduction, thus allowing negative processing times and non infinite c_{ij} values on $\mathbb{R}^{*-}$. Nagata et al. (2010) and Hashimoto et al. (2008) proposed forward and backward dynamic programming functions, which allowed to concatenate only pairs of subsequences. We present here the re-optimization approach of Vidal et al. (2011), which enables to work with any number of concatenations, and accounts for duration features. This approach is closely related to the method of Kindervater and Savelsbergh (1997b) for $\{\emptyset|DUR\}$ and $\{\emptyset|DUR, TW\}$.

Data: Minimum duration $D(A_i)$ to perform all the activities of A_i but the last one. Earliest execution date $E(A_i)$ of the first activity in any feasible schedule with minimum idle time for A_i. Latest possible execution date $L(A_i)$ of the first activity in any feasible schedule with minimum processing time reduction for A_i. Minimum processing time reduction $TW(A_i)$ in any feasible schedule for A_i.

Data computation and evaluate concatenation: For a sequence A with a single activity, we have $D(A) = TW(A) = 0$, $E(A) = e_{A(1)}$ and $L(A) = l_{A(1)}$. Equations (52-57) enable both to

evaluate the cost of concatenation and compute the data for sequences with more activities:

$$\text{with } \delta_{WT} = \max\{E(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|)A_2(1)} - L(A_1), 0\} \tag{52}$$

$$\text{and } \delta_{TW} = \max\{E(A_1) + D(A_1) - TW(A_1) + p_{A_1(|A_1|)A_2(1)} - L(A_2), 0\} \tag{53}$$

$$E(A_1 \oplus A_2) = \max\{E(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|)A_2(1)}, E(A_1)\} - \delta_{WT} \tag{54}$$

$$L(A_1 \oplus A_2) = \min\{L(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|)A_2(1)}, L(A_1)\} + \delta_{TW} \tag{55}$$

$$D(A_1 \oplus A_2) = D(A_1) + D(A_2) + p_{A_1(|A_1|)A_2(1)} + \delta_{WT} \tag{56}$$

$$TW(A_1 \oplus A_2) = TW(A_1) + TW(A_2) + \delta_{TW} \tag{57}$$

This approach enables to evaluate in $O(1)$ the minimum amount of processing time reduction for any constant number of subsequence concatenations. In contrast, soft time-window relaxations $\{TW|P\}$ or $\{D|R,P\}$ only allowed for $O(\log n)$ move evaluations.

General flexible processing-times. Sourd (2005) and Hashimoto et al. (2006) tackle a more general serial timing problem with flexible and sequence dependent processing time/cost trade-off functions, and costs on activity execution dates $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$. This problem is formulated in Equation 58. Functions $c_i(t)$ and $c'_{ij}(t)$ are assumed to be piecewise linear, lower semicontinuous, non negative and take infinite value for $t < 0$, and functions $c'_{\sigma_i\sigma_{i+1}}(\Delta t)$ are convex.

$$\min_{t_1 \ldots t_n} \sum_{i=1}^{n} c_i(t_i) + \sum_{i=1}^{n-1} c'_{\sigma_i\sigma_{i+1}}(t_{\sigma_{i+1}} - t_{\sigma_i}) \tag{58}$$

Data: Optimal cost $F(A_i)(t)$ (respectively $B(A_i)(t)$) of a schedule for A_i, when the first (respectively the last) activity is executed exactly at t.

Data computation: For a sequence A with a single activity, $F(A)(t) = B(A)(t) = c_{A(1)}(t)$. Equations (59-60) can then be used to compute and store the $F(A_i)(t)$ and $B(A_i)(t)$ functions respectively on prefix and suffix subsequences of the incumbent timing problem.

$$F(A_i \oplus A)(t) = c_{A(1)}(t) + \min_{0 \le x \le t}\{F(A_i)(x) + c'_{A_i(|A_i|)A(1)}(t-x)\} \tag{59}$$

$$B(A \oplus A_i)(t) = c_{A(1)}(t) + \min_{x \ge t}\{B(A_i)(x) + c'_{A(1)A_i(|A_i|)}(x-t)\} \tag{60}$$

Evaluate concatenation: Equation (61) returns the optimal cost $Z^*(A_1 \oplus A_2)$ of the timing problem related to the concatenation of a prefix and a suffix subsequence.

$$Z^*(A_1 \oplus A_2) = \min_{t}\{F(A_1(t) + \min_{t'}(c'_{A_1(|A_1|)A_2(1)}(t'-t) + B(A_2)(t')\} \tag{61}$$

Evaluating the concatenation of two subsequences can be performed in $O(\varphi_c + \widehat{\varphi_c} \times \varphi'_c)$ (Hashimoto et al. 2006), where φ_c and φ'_c are respectively the total number of pieces in cost functions c_i and $c'_{\sigma_i\sigma_{i+1}}$, and $\widehat{\varphi_c}$ represents the number of convex pieces in $c'_{\sigma_i\sigma_{i+1}}$.

Finally, we remarked in Section 6.3 that $\{DUR|MTW\}$ and $\{\emptyset|DUR, MTW\}$ constitute special cases of $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$. Hence, the previous re-optimization approach applies, leading to an amortized complexity of $O(\varphi_{\text{MTW}})$ for solving successive $\{DUR|MTW\}$ or $\{\emptyset|DUR, MTW\}$ subproblems, where φ_{MTW} denotes the total number of time windows in the problem,

and when the number of subproblems is large enough. This consequence leads to theoretical (and eventually practical) improvements upon the previous procedures in $O(n\varphi_{\text{MTW}})$ for these settings.

Time-dependent processing times. Time-dependent processing times, when coupled with time-window constraints or other activity costs, lead to complicated timing problems, which often require additional assumptions on the nature of the processing times, such as the FIFO or HYI rules (Section 6.4), to ensure polynomial solution methods.

Donati et al. (2008) studied the feasibility problem $\{\emptyset|P(t),TW\}$ within local searches for vehicle routing, and proposed to extend the methodology of Savelsbergh (1985) for time-windows feasibility checking to time-dependent processing times. Let $g_{ij}(t) = t + p_{ij}(t)$ be defined as the completion date of an activity i started at t, and followed by activity j. Under the assumption that all $g_{ij}(t)$ are continuous and strictly increasing (any activity started strictly later will finish strictly later), the inverse function $g_{ij}^{-1}(t)$ can be defined, and the following re-optimization method enables to perform efficient feasibility checking:

Data: Earliest possible execution date $E(A_i)$ of the last activity in any feasible schedule for A_i. Latest possible execution date $L(A_i)$ of the first activity in any feasible schedule for A_i.

Data computation: For a sequence A with a single activity, $E(A) = e_{A(1)}$, and $L(A) = l_{A(1)}$. Equations (62-65) then enables to determine the re-optimization data on prefix and suffix subsequences, by forward and backward dynamic programming.

$$E(A_1 \oplus A) = \max\{e_{A(1)}, g(E(A_1))\} \tag{62}$$

$$isFeas(A_1 \oplus A) \equiv isFeas(A_1) \ \wedge \ E(A_1 \oplus A) \leq l_{A(1)} \tag{63}$$

$$L(A \oplus A_2) = \min\{l_{A(1)}, g^{-1}(L(A_2))\} \tag{64}$$

$$isFeas(A \oplus A_2) \equiv isFeas(A_2) \ \wedge \ L(A \oplus A_2) \geq e_{A(1)} \tag{65}$$

Evaluate concatenation: Equation (66) enables to state on the feasibility of any concatenation of a pair of prefix and suffix subsequences.

$$isFeas(A_1 \oplus A_2) \equiv isFeas(A_1) \wedge isFeas(A_2) \wedge E(A_1) + p_{A_1(|A_1|)A_2(1)}(E(A_1)) \leq L(A_2) \tag{66}$$

Under the assumption that function $g^{-1}(t)$ is evaluated in $O(1)$, the previous re-optimization framework enables to evaluate the feasibility of a concatenation of two subsequences in $O(1)$. However, it does not allow to concatenate more than two subsequences, as opposed to the fixed processing-time setting treated in Section 8.5. For these cases, particular orders of timing problem evaluations can eventually enable to solely rely on forward extension, backward extension and concatenation of two subsequences to perform re-optimization.

For a general case with time-dependent (and sequence dependent) processing times with separable execution costs $\{\Sigma c_i(t_i)|P(t)\}$, encompassing the previous problems as a special case, Hashimoto et al. (2008) proposed a dynamic programming approach is the same spirit as those of Sections (8.5) and (8.6). Concatenation evaluations for pairs of subsequences A_1 and A_2 can then be performed in $O(\varphi_c + \varphi_p)$, where φ_c and φ_p denote respectively the total number of pieces in the cost and processing-time functions of the subproblem at play. For the sake of brevity, details of the related re-optimization operators are given in the Appendix C.

Table 3: Complexity of algorithms and re-optimization operators for common timing problems

Problem	From Scratch		Re-opt. by concat.	F/B	C2	C3+	Sd	Assumptions
Const. act. costs	—	$O(n)$	—	$O(1)$	$O(1)$	$O(1)$	✓	
$\{W \circ\}$	Min idle time	$O(n)$	—	$O(1)$	$O(1)$	$O(1)$	✓	
$\{\circ\, TW\}$	Min idle time	$O(n)$	Savelsbergh (1985) & Kind. and Sav. (1997)	$O(1)$	$O(1)$	$O(1)$	✓	
$\{D \circ\}$	Min idle time	$O(n)$	Ergun and Orlin (2006)	$O(\log n)$	$O(1)$*	—		penalty coefficient depending upon act.
$\{D, R(d_i = r_i).NWT\}$	Min idle time	$O(n)$	Kedad-Sidhoum and Sourd (2010)	$O(\log n)$	$O(1)$*	—		penalty coefficient depending upon act.
$\{D, R(d_i = r_i) \circ\}$	Garey et al. (1988) & Ahuja and Orlin (2001)	$O(n \log n)$	Ibaraki et al. (2008)	$O(\log n)$		$O(\log n)$	✓	
$\{D\, R\}$	Min idle time	$O(n)$	Ibaraki et al. (2008)	$O(\log n)$		$O(\log n)$	✓	
$\{\sum c_i^{CX}(t_i) \circ\}$	Ibaraki et al. (2008)	$O(n \log \tilde{\varphi}_c)$	Ibaraki et al. (2008)	$O(\log \tilde{\varphi}_c)$		$O(\log \tilde{\varphi}_c)$	✓	cost f. ≥ 0, p.l. & l.s.c
$\{\sum c_i(t_i) \circ\}$	Ibaraki et al. (2005)	$O(n\tilde{\varphi}_c)$	Ibaraki et al. (2005)	$O(\tilde{\varphi}_c)$		$O(\tilde{\varphi}_c)$	✓	cost f. ≥ 0, p.l. & l.s.c
$\{\circ\, MTW\}$	Min idle time	$O(n - \tilde{\varphi}nrw)$	Ibaraki et al. (2005)	$O(\log \tilde{\varphi}nrw)$		—	✓	
$\{DUR\,TW\}$, $\{\circ\, DUR.TW\}$	Malcolm et al. (1959)	$O(n)$	Savelsbergh (1992) & Kind. and Sav. (1997)	$O(1)$		$O(1)$	✓	
$\{DUR.MTW\}$, $\{\circ\, DUR.MTW\}$	Tricoire et al. (2010)	$O(n\tilde{\varphi}MTW)$	Hashimoto et al. (2006)	$O(\tilde{\varphi}nrw)$		—	✓	
$\{\circ\, IDL.TW\}$	Hunsaker and S. (2002)	$O(n)$	—	—	—	—		
$\{\sum c_i^{CX}(\Delta t_i).\Sigma c_i(t_i) \circ\}$	Sourd (2005) & Hashimoto et al. (2006)	$O(n(\tilde{\varphi}_c - \widehat{\tilde{\varphi}_c} \times \tilde{\varphi}_c))$	Sourd (2005) & Hashimoto et al. (2006)	$O(\tilde{\varphi}_c - \widehat{\tilde{\varphi}_c} \times \tilde{\varphi}_c)$		—	✓	cost f. ≥ 0, p.l. & l.s.c
$\{D\, R.P(t)\}$	Min idle time	$O(n)$	—	—	—	—		FIFO assumption
$\{\circ\, TW.P(t)\}$	Dozari et al. (2008)	$O(n)$	Dozari et al. (2008)	$O(1)$		—	✓	FIFO assumption
$\{\Sigma c_i(t_i).P(t)\}$	Hashimoto et al. (2008)	$O(n(\tilde{\varphi}_c - \tilde{\varphi}_p))$	Hashimoto et al. (2008)	$O(\tilde{\varphi}_c - \tilde{\varphi}_p)$		—	✓	cost f. ≥ 0, p.l. & l.s.c & HYI assumption
$\{\circ\, TL.TW\}$	Hurink and Keuchel (2001)	$O(n^3)$	—	—	—	—		
$\{\circ\, TL.TW\}$	Haugland and Ho (2010)	$O(n \log n)$	—	—	—	—		$O(n)$ TL constraints
$\{DUR > D > TL.R\}$	Cordeau and Laporte (2003)	$O(n^2)$	—	—	—	—		$O(n)$ TL constraints & LIFO assumption
$\{\sum c_i^{CX}(t_i - t_i).\Sigma c_i^{CX}(t_i) \circ\}$	Ahuja et al. (2003)	$O(n^3 \log n \log(nU))$	—	—	—	—		U is an upper bound of execution dates

8.7 Perspectives: re-optimization for two-dimension features

Table 3 summarizes complexities of stand-alone and re-optimization approaches for the main timing settings in the literature. The leftmost column lists the problems, while the next block of columns present stand-alone approaches and their complexity. Following to the right, the next columns are dedicated to summarize re-optimization approaches, the complexity of forward and backward data construction, "F/B", the complexity of concatenation of two, "C2", and more than two subsequences, "C3+", if available. Column "Sd" finally indicates whether the re-optimization approach enables to tackle problems with sequence dependent parameters. Additional assumptions on the problems are listed in the last column.

Re-optimization has clearly proven successful to tackle several settings more efficiently than from scratch, but still results in several challenges regarding two-dimension features. Several problems remain, such as $\{DUR|MTW\}$, $\{\emptyset|DUR, MTW\}$, $\{DUR|TW, P(t)\}$, $\{\emptyset|TL, TW\}$ and $\{DUR > D > TL|R\}$, for which serial timing has not actually be considered. Another research issue involves the study of re-optimization data and concatenation operators involving more than two subsequences, and able to manage sequence dependent problems. Such operators are still missing in several cases, such as $\{\Sigma c_i(t_i)|P(t)\}$ or $\{\Sigma c_i^{TVS}(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$, for which particular evaluation orders of timing subproblems are for now necessary.

9 Conclusion

We described and classified a rich body of problems with time characteristics and totally ordered variables. Methods from various fields of research have been analyzed, in order to present key problem features and main solving concepts. As timing subproblems frequently arise in the context of local searches, we analyzed both stand-alone resolution, and efficient solving of series of problems by means of re-optimization approaches. A general re-optimization framework, based on decomposition and recombination of sequences, has been introduced to that extent, and links with actual re-optimization approaches have been examined. We identified a subset of timing methods originating from various research fields, constituting the actual state-of-the-art for main timing problems. For several combinatorial optimization settings involving timing subproblems, this development is the key to progress towards more generic solvers, relying on the algorithms presented in this paper to tackle a wide range of problem particularities.

As a result of this work, many interesting avenues of research arise. First of all, for many timing features studied in this article, more efficient stand-alone or re-optimization methods should be sought, and dedicated work would also be conducted on better exploiting the particularities of problems at hand, such as the nature of permutations functions for serial timing. The impact of sequence-dependency on re-optimization is another interesting concern, as sequence-dependency constitute a fundamental delimitation between routing related problems and scheduling settings. Identifying precisely its impact on re-optimization procedures should lead to a better insight on local search based methods for these two important classes of problems. Finally, even if the focus has been put on time characteristics, other cumulative resources such as loads, energy or workforce lead to similar features and models. The work performed on timing can thus prove useful for an even broader range of applications.

Acknowledgments

While working on this project, T.G. Crainic was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway, M. Gendreau was the NSERC/Hydro-Québec Industrial Research Chair on the Stochastic Optimization of Electricity Generation, MAGI, École Polytechnique, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal.

Partial funding for this project has been provided by the Champagne-Ardennes regional council, France, the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by our partners CN, Rona, Alimentation Couche-Tard, la Fédération des producteurs de lait du Québec, the Ministry of Transportation of Québec, and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) through its Team Research Project program.

References

Ahuja, R, D Hochbaum, J Orlin. 2003. Solving the convex cost integer dual network flow problem. *Management Science* **49**(7) 950–964.

Ahuja, Ravindra K., James B. Orlin. 2001. A Fast Scaling Algorithm for Minimizing Separable Convex Functions Subject to Chain Constraints. *Operations Research* **49**(5).

Alidaee, B., N.K. Womer. 1999. Scheduling with time dependent processing times: review and extensions. *Journal of the Operational Research Society* **50**(7) 711–720.

Archetti, C., M. Savelsbergh. 2009. The trip scheduling problem. *Transportation Science* **43**(4) 417–431.

Ayer, M., H.D. Brunk, G.M. Ewing, W.T. Reid, E. Silverman. 1955. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics* **26**(4) 641–647.

Baker, K.R., G.D. Scudder. 1990. Sequencing with earliness and tardiness penalties: a review. *Operations Research* **38**(1) 22–36.

Balakrishnan, N. 1993. Simple heuristics for the vehicle routeing problem with soft time windows. *Journal of the Operational Research Society* **44**(3) 279–287.

Baldacci, R., A. Mingozzi, R. Roberti. 2011. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research - Forthcoming* .

Barlow, R.E., D.J. Bartholomew, J.M. Bremner, H.D. Brunk. 1972. *Statistical inference under order restrictions: The theory and application of isotonic regression*. Wiley New York.

Bartusch, M., R.H. Möhring, F.J. Radermacher. 1988. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* **16**(1) 199–240.

Bean, J.C., J.R. Birge, R.L. Smith. 1987. Aggregation in dynamic programming. *Operations Research* **35**(2) 215–220.

Beasley, J.E. 1981. Adapting the savings algorithm for varying inter-customer travel times. *Omega* **9**(6) 658–659.

Berbeglia, G., J.-F. Cordeau, G. Laporte. 2010. A Hybrid Tabu Search and Constraint Programming Algorithm for the Dynamic Dial-a-Ride Problem. *CIRRELT Working Paper* .

Best, M.J., N. Chakravarti. 1990. Active set algorithms for isotonic regression, a unifying framework. *Mathematical Programming* **47**(1-3) 425–439.

Best, M.J, N. Chakravarti, V.A. Ubhaya. 2000. Minimizing separable convex functions subject to simple chain constraints. *SIAM Journal on Optimization* **10**(3) 658–672.

Bianco, L., A. Mingozzi, S. Ricciardelli. 1993. The traveling salesman problem with cumulative costs. *Networks* **23**(2) 81–91.

Blum, A., P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan. 1994. The minimum latency problem. *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM, New York, USA, 163–171.

Bräysy, O., M. Gendreau. 2005a. Vehicle Routing Problem with Time Windows , Part II : Metaheuristics. *Transportation Science* **39**(1) 119–139.

Bräysy, O., M. Gendreau. 2005b. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science* **39**(1) 104–118.

Brucker, P, A Drexl, R Möhring, K Neumann, E. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operations Research* **112**(1) 3–41.

Brunk, H. D. 1955. Maximum Likelihood Estimates of Monotone Parameters. *The Annals of Mathematical Statistics* **26**(4) 607–616.

Campbell, A.M., M. Savelsbergh. 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation science* **38**(3) 369–378.

Chakravarti, N. 1989. Isotonic Median Regression: A Linear Programming Approach. *Mathematics of Operations Research* **14**(2) 303–308.

Cheng, T. 2004. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research* **152**(1) 1–13.

Chrétienne, P., F. Sourd. 2003. PERT scheduling with convex cost functions. *Theoretical Computer Science* **292**(1) 145–164.

Christiansen, M., K. Fagerholt. 2002. Robust ship scheduling with multiple time windows. *Naval Research Logistics* **49**(6) 611–625.

Cooke, K.L., E. Halsey. 1966. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications* **14** 493–498.

Cordeau, J.-F., G. Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B* **37**(6) 579–594.

Cordeau, J.-F., G. Laporte, A. Mercier. 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* **52**(8) 928–936.

Cordeau, J.-F., G. Laporte, A. Mercier. 2004. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society* **55**(5) 542–546.

Cordone, R. 2000. A note on time windows constraints in routing problems. *Tech. Report, Department of Electronics and Information, Polytechnic of Milan* .

Davis, J.S., J.J. Kanet. 1993. Single-machine scheduling with early and tardy completion costs. *Naval Research Logistics* **40** 85–101.

Dechter, R, I. Meiri, J. Pearl. 1991. Temporal constraint networks. *Artificial Intelligence* **49** 61–95.

Desaulniers, G., J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, D. Villeneuve. 1998. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. T.G. Crainic, G. Laporte, eds., *Fleet Management and Logistics*. Kluwer Academic Publishers, Norwell, MA, 129–154.

Desaulniers, G., J. Desrosiers, M.M. Solomon, eds. 2005. *Column generation*. Springer.

Desaulniers, G., D. Villeneuve. 2000. The Shortest Path Problem with Time Windows and Linear Waiting Costs. *Transportation Science* **34**(3) 312–319.

Desrochers, J.K., J.K. Lenstra, M.W.P. Savelsbergh. 1990. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research* **46**(3) 322–332.

Desrochers, M, J Desrosiers, M M Solomon. 1992. A new optimization algorithm for the Vehicle-Routing problem with time windows. *Operations Research* **40**(2) 342–354.

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M.O. Ball, C.L. Monma, G.L. Nemhauser, eds., *Handbooks in operations research and management science, Volume 8: Network Routing*. Elsevier, Amsterdam, 35–139.

Donati, A., R. Montemanni, N. Casagrande, A.E. Rizzoli, L.M. Gambardella. 2008. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research* **185**(3) 1174–1191.

Dreyfus, SE. 1969. An appraisal of some shortest-path algorithms. *Operations Research* **17**(3) 395–412.

Dumas, Y, F Soumis, J Desrosiers. 1990. Optimizing the Schedule for a Fixed Vehicle Path with Convex Inconvenience Costs. *Transportation Science* **24**(2) 145–152.

Elhallaoui, I., D. Villeneuve, F. Soumis, G. Desaulniers. 2005. Dynamic Aggregation of Set-Partitioning Constraints in Column Generation. *Operations Research* **53**(4) 632–645.

Ergun, O., J. Orlin. 2006. Fast neighborhood search for the single machine total weighted tardiness problem. *Operations Research Letters* **34**(1) 41–45.

Erkut, E., J. Zhang. 1996. The Maximum Collection Problem with Time-Dependent Rewards. *Naval Research Logistics* **43**(5) 749–763.

Feillet, D., P. Dejax, M. Gendreau. 2005. Traveling Salesman Problems with Profits. *Transportation Science* **39**(2) 188–205.

Feng, G., H.C. Lau. 2007. Efficient algorithms for machine scheduling problems with earliness and tardiness penalties. *Annals of Operations Research* **159**(1) 83–95.

Fischetti, M., G. Laporte, S. Martello. 1993. The Delivery Man Problem and Cumulative Matroids. *Operations Research* **41**(6) 1055–1064.

Fleischmann, B., M. Gietz, S. Gnutzmann. 2004. Time-Varying Travel Times in Vehicle Routing. *Transportation Science* **38**(2) 160–173.

Frangioni, A., A. Manca. 2006. A Computational Study of Cost Reoptimization for Min-Cost Flow Problems. *INFORMS Journal on Computing* **18**(1) 61–70.

Fredman, M.L. 1975. On computing the length of longest increasing subsequences. *Discrete Mathematics* **11**(1) 29–35.

Garey, M.R., R.E. Tarjan, G.T. Wilfong. 1988. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research* **13**(2) 330–348.

Gawiejnowicz, S. 2008. *Time-dependent scheduling*. Springer-Verlag New York Inc.

Gendreau, M., C.D. Tarantilis. 2010. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. *CIRRELT Working Paper* .

Giffler, B., G. L. Thompson. 1960. Algorithms for Solving Production-Scheduling Problems. *Operations Research* **8**(4) 487–503.

Goel, A. 2010. Truck driver scheduling in the European Union. *Transportation Science* **44**(4) 429–441.

Goel, A., L. Kok. 2009. Efficient truck driver scheduling in the United States. *Working Paper, University of Twente* (561).

Goto, S., A. Sangiovanni-Vincentelli. 1978. A new shortest path updating algorithm. *Networks* **8**(4) 341–372.

Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* **5** 287–326.

Grotzinger, S.J., C. Witzgall. 1984. Projections onto Order Simplexes. *Applied Mathematics and Optimization* **270**(12) 247–270.

Halpern, J. 1977. Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Zeitschrift für Operations Research* **21**(3) 117–124.

Hashimoto, H. 2008. Studies on Local Search-Based Approaches for Vehicle Routing and Scheduling Problems. Ph.D. thesis, Kyoto University.

Hashimoto, H., T. Ibaraki, S. Imahori, M. Yagiura. 2006. The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics* **154**(16) 2271–2290.

Hashimoto, H., M. Yagiura, T. Ibaraki. 2008. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization* **5**(2) 434–456.

Hashimoto, H., M. Yagiura, S. Imahori, T. Ibaraki. 2010. Recent progress of local search in handling the time window constraints of the vehicle routing problem. *4or* **8**(3) 221–238.

Haugland, D., S. Ho. 2010. Feasibility Testing for Dial-a-Ride Problems. *Proceedings of the 6th international conference on Algorithmic aspects in information and management, AAIM'10, Weihai, China*. Springer-Verlag Berlin Heidelberg, 170–179.

Hendel, Y., F. Sourd. 2006. Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem. *European Journal of Operational Research* **173**(1) 108–119.

Hendel, Y., F. Sourd. 2007. An improved earliness-tardiness timing algorithm. *Computers & Operations Research* **34**(10) 2931–2938.

Hochbaum, D. 2002. Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research* **140**(2) 291–321.

Hochbaum, D.S., J.G. Shanthikumar. 1990. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM (JACM)* **37**(4) 843–862.

Hoogeveen, J.A., S.L. Van De Velde. 1996. A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing* **8**(4) 402–412.

Hoos, H.H., T. Stützle. 2005. *Stochastic local search: Foundations and applications*. Morgan Kaufmann.

Hunsaker, B., M.W.P. Savelsbergh. 2002. Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters* **30**(3) 169–173.

Hurink, J., J. Keuchel. 2001. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics* **112**(1-3) 179–197.

Ibaraki, T., S. Imahori, M. Kubo, T. Masuda, T. Uno, M. Yagiura. 2005. Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints. *Transportation Science* **39**(2) 206–232.

Ibaraki, T., S. Imahori, K. Nonobe, K. Sobue, T Uno, M Yagiura. 2008. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics* **156**(11) 2050–2069.

Ichoua, S., M. Gendreau, J.Y. Potvin. 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* **144**(2) 379–396.

Ioachim, I., S. Gélinas, F. Soumis, J. Desrosiers. 1998. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks* **31**(3) 193–204.

Irnich, S. 2008a. A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics. *INFORMS Journal on Computing* **20**(2) 270–287.

Irnich, S. 2008b. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum* **30** 113–148.

Kallehauge, B., J. Larsen, O.B.G. Madsen, M.M. Solomon. 2005. Vehicle routing problem with time windows. G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., *Column Generation*. Springer, 67–98.

Kanet, J.J., V Sridharan. 2000. Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research* **48**(1) 99–110.

Karmarkar, N. 1984. A New Polynomial-Time algorithm for Linear Programming. *Combinatorica* **4**(4) 373–395.

Kedad-Sidhoum, S., F. Sourd. 2010. Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers & Operations Research* **37**(8) 1464–1471.

Kelley, J.E., M.R. Walker. 1959. Critical-path planning and scheduling. *Proceedings of Eastern joint Computer conference*. ACM Press, New York, New York, USA, 160–173.

Khachiyan, L.G. 1979. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady* **20** 191–194.

Kindervater, G.A.P., M.W.P. Savelsbergh. 1997a. Vehicle routing: Handling edge exchanges. E. Aarts, J.K. Lenstra, eds., *Local Search in Combinatorial Optimization*. Wiley, 337–360.

Kindervater, G.A.P., M.W.P. Savelsbergh. 1997b. Vehicle routing: Handling edge exchanges. *Local Search in Combinatorial Optimization* 337–360.

Kok, A.L., E.W. Hans, J.M.J. Schutten. 2009. Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Working Paper, University of Twente* 1–25.

Kok, A.L., C.M. Meyer, H. Kopfer, J.M.J. Schutten. 2010. A dynamic programming heuristic for the vehicle routing problem with time windows and European Community social legislation. *Transportation Science* **44**(4) 442–454.

Koskosidis, Y A, W B Powell, M M Solomon. 1992. An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints. *Transportation Science* **26**(2) 69–85.

Lee, C.Y., J.Y. Choi. 1995. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research* **22**(8) 857–869.

Malandraki, C., M.S. Daskin. 1992. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science* **26**(3) 185–200.

Malandraki, C., R.B. Dial. 1996. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research* **90**(1) 45–55.

Malcolm, D.G., J.H. Roseboom, C.E. Clark, W. Fazar. 1959. Application of a technique for research and development program evaluation. *Operations Research* **7**(5) 646–669.

Miller-Hooks, E., B. Yang. 2005. Updating Paths in Time-Varying Networks Given Arc Weight Changes. *Transportation Science* **39**(4) 451–464.

Mitten, L.G. 1959. Sequencing n Jobs on Two Machines with Arbitrary Time Lags. *Management Science* **5**(3) 293–298.

Nagata, Y. 2007. Effective memetic algorithm for the vehicle routing problem with time windows: Edge assembly crossover for the VRPTW. *Proceedings of the Seventh Metaheuristics International Conference, Montreal, Canada (on CD-ROM)*.

Nagata, Y., O. Bräysy, W. Dullaert. 2010. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* **37**(4) 724–737.

Ngueveu, S.U., C. Prins, R. Wolfler Calvo. 2010. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research* **37**(11) 1877–1885.

Norstad, I., K. Fagerholt, G. Laporte. 2010. Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies, Forthcoming* .

Or, I. 1976. Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Northwestern University, Evanston, IL.

Pallottino, S. 2003. A new algorithm for reoptimizing shortest paths when the arc costs change. *Operations Research Letters* **31**(2) 149–160.

Pan, Y., L. Shi. 2005. Dual Constrained Single Machine Sequencing to Minimize Total Weighted Completion Time. *IEEE Transactions on Automation Science and Engineering* **2**(4) 344–357.

Pardalos, P. 1995. Efficient computation of an isotonic median regression. *Applied Mathematics Letters* **8**(2) 67–70.

Pardalos, P.M., G Xue. 1999. Algorithms for a class of isotonic regression problems. *Algorithmica* **23**(3) 211–222.

Parragh, S.N., J.-F. Cordeau, K.F. Doerner, R.F. Hartl. 2010. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum* .

Patriksson, M. 2008. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research* **185**(1) 1–46.

Pinedo, M. 2008. *Scheduling: theory, algorithms, and systems*. Prentice Hall.

Potts, C.N., J.D. Whitehead. 2007. Heuristics for a coupled-operation scheduling problem. *Journal of the Operational Research Society* **58**(10) 1375–1388.

Potvin, J.-Y., J.-M. Rousseau. 1995. An Exchange Heuristic for Routeing Problems with Time Windows. *The Journal of the Operational Research Society* **46**(12) 1433–1446.

Prescott-Gagnon, E., G. Desaulniers, M. Drexl, L.-M. Rousseau. 2010. European Driver Rules in Vehicle Routing with Time Windows. *Transportation Science* **44**(4) 455–473.

Prescott-Gagnon, E., Guy Desaulniers, L.M. Rousseau. 2009. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks* **54**(4) 190–204.

Private communication. 2010.

Righini, G, M Salani. 2006. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* **3**(3) 255–273.

Robertson, T, F T Wright, R L Dykstra. 1988. *Order Restricted Statistical Inference (Wiley Series in Probability and Statistics)*. John Wiley & Sons.

Rockafellar, R.T. 1970. *Convex analysis*. Princeton Univ Pr.

Roy, B. 1959. Contribution de la théorie des graphes à l'étude de certains problèmes linéaires. *Comptes rendus de l'académie des sciences de Paris* **248** 2437–2439.

Roy, B. 1962. Graphes et ordonnancement. *Revue Française de Recherche Opérationnelle* **25** 323–333.

Savelsbergh, M.W.P. 1985. Local Search in Routing Problems with Time Windows. *Annals of Operations Research* **4**(1) 285–305.

Savelsbergh, M.W.P. 1992. The Vehicle Routing Problem with Time Windows: Minimizing Route Duration. *ORSA Journal on Computing* **4**(2) 146–154.

Sexton, T.R., L.D. Bodin. 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transportation Science* **19**(4) 378–410.

Sexton, T.R., L.D. Bodin. 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. *Transportation Science* **19**(4) 411–435.

Solomon, M.M., J. Desrosiers. 1988. Time Window Constrained Routing and Scheduling Problems. *Transportation Science* **22**(1) 1–13.

Sourd, F. 2005. Optimal timing of a sequence of tasks with general completion costs. *European Journal of Operational Research* **165**(1) 82–96.

Sourd, F., S. Kedad-Sidhoum. 2003. The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling* **6**(6) 533–549.

Szwarc, W., S.K. Mukhopadhyay. 1995. Optimal timing schedules in earliness-tardiness single machine sequencing. *Naval Research Logistics* **42**(7) 1109–1114.

Tagmouti, M., M. Gendreau, J.-Y. Potvin. 2007. Arc routing problems with time-dependent service costs. *European Journal of Operational Research* **181**(1) 30–39.

Taillard, E., P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin. 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science* **31**(2) 170–186.

Talbot, F.B. 1982. Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case. *Management Science* **28**(10) 1197–1210.

Tang, J., Y. Kong, H. Lau, A.W.H. Ip. 2010. A note on Efficient feasibility testing for dial-a-ride problems. *Operations Research Letters* **38**(5) 405–407.

Tricoire, F., M. Romauch, K.F. Doerner, R.F. Hartl. 2010. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research* **37**(2) 351–367.

Van Woensel, T., L. Kerbache, H. Peremans, N. Vandaele. 2008. Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research* **186**(3) 990–1007.

Vidal, T., T.G. Crainic, M. Gendreau. 2011. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows.

Walshaw, C. 2002. A multilevel approach to the travelling salesman problem. *Operations Research* **50**(5) 862–877.

Wan, G., B.P.-C. Yen. 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research* **142**(2) 271–281.

Yano, C.A., Y.-D. Kim. 1991. Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operational Research* **52**(2) 167–178.

Appendix

A Reduction of LISP to $\{TW(unit)|P\}$

Given a vector $\mathbf{N} = (N_1 \dots N_n)$ of n real numbers, LISP aims to find the maximum length L of a non-decreasing subsequence of numbers: $L = max\{k : 1 \leq i_1 < \dots < i_k \leq n \text{ and } N_{i_1} \leq \dots \leq N_{i_k}\}$.

From a LISP instance, we construct the following instance $\mathcal{T}$ of $\{TW(unit)|\emptyset\}$, with n activities such that for all $i \in [\![1, n]\!]$, $c_i = l_i = N_i$ and $p_i = 0$. This instance is created in n elementary algorithmic operations.

Let $z^*(\mathcal{T})$ be the optimal solution cost of $\mathcal{T}$. This solution naturally initiates as much activities as possible without penalties, within a non-decreasing order of execution dates. Hence, the activities realized without penalties corresponds to the LISP subsequence sought in the original problem, whose length is $L^* = n - z^*(\mathcal{T})$. Hence, LISP admits a many-one reduction to $\{TW(unit)|\emptyset\}$. Conversely, $\{TW(unit)|\emptyset\}$ generalizes LISP.

B Proof of Theorem 1: Block optimality conditions of $\{\Sigma c_i^{\mathbf{cvx}}(t_i)|\emptyset\}$

We first recall the timing problem at play, given in Equations (67-68).

$$\min_{(t_1 \dots t_n)} \quad \sum_{i=1}^{n} c_i(t_i) \tag{67}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{68}$$

The functions $c_i(t) : \Re \to \Re$ are assumed to be convex, but not necessarily smooth. We denote as $\delta c_i(t)$ the subdifferential of c_i at t, which is necessarily a non-empty interval, as a byproduct of the convexity assumption and the space of definition. We first recall some useful properties on subdifferentials, to follow with the proof of Theorem (1).

Proposition 2. *Let $f_1 \dots f_m$ be subdifferentiable functions on $\Re^n$, then:*

$$\delta(f_1(x) + \dots + f_m(x)) \supset \delta f_1(x) + \dots + \delta f_m(x) \; \forall x \tag{69}$$

Theorem 2 (Rockafellar 1970). *Let $f_1 \dots f_m$ be a set of proper convex functions on $\Re^n$ having at least one common point in the relative interior of their domains $ri(dom(f_i))$, then:*

$$\delta(f_1(x) + \dots + f_m(x)) = \delta f_1(x) + \dots + \delta f_m(x) \; \forall x \in \bigcap ri(dom(f_i)) \tag{70}$$

Constraint qualifications hold as the constraints are linear, and any solution with idle time between each activity is feasible, thus taking place in the relative interior of the polytope. Hence, strong duality applies, leading to the following necessary and sufficient optimality conditions: A solution $\mathbf{t}^* = (t_1^* \dots t_n^*)$ of Problem (67-68) is optimal if and only if a set of Lagrangian multipliers $(\lambda_1^* \dots \lambda_{n-1}^*)$ exists, such that conditions (71) are satisfied (Bertsekas2003 , Propositions 5.7.1 and 6.4.2).

$$\begin{cases} t^*_{i-1} + p_{i-1} \leq t^*_i & \forall i \in [\![2, n]\!] \\ 0 \in \delta c_1(t^*_1) + \lambda^*_1 & \\ 0 \in \delta c_i(t^*_i) + \lambda^*_i - \lambda^*_{i-1} & \forall i \in [\![2, n-1]\!] \\ 0 \in \delta c_n(t^*_n) - \lambda^*_{n-1} & \\ \lambda^*_{i-1}(t^*_{i-1} + p_{i-1} - t^*_i) = 0 & \forall i \in [\![2, n]\!] \\ \lambda^*_i \geq 0 & \forall i \in [\![1, n]\!] \end{cases} \tag{71}$$

Solution $\mathbf{t}^*$ can be represented as a succession of blocks of activities $(B_1 \dots B_m)$, such that activities within each block are processed without idle time, and last activities of blocks are followed by non-null idle time. The previous definition, combined with primal feasibility, involves that $t^*_{B_j(|B_j|)} + p_{B_j(|B_j|)} < t^*_{B_j(|B_j|)+1}$, and thus $\lambda^*_{B_j(|B_j|)} = 0$ for each $j \in [\![1, m-1]\!]$. Conditions (71) are thus equivalent to primal feasibility (equivalent to the first condition of (1) combined with the definition of blocks) and the following independent systems of equations for each block:

$$\forall j \in [\![1, m]\!], \begin{cases} \textbf{i)} & 0 \in \delta c_{B_j(1)}(t^*_{B_j(1)}) + \lambda^*_{B_j(1)} & \\ \textbf{ii)} & 0 \in \delta c_i(t^*_{B_j(1)} + p_{B_j(1)i-1}) + \lambda^*_i - \lambda^*_{i-1} & \forall i \in [\![B_j(1)+1, B_j(|B_j|)-1]\!] \\ \textbf{iii)} & 0 \in \delta c_{B_j(|B_j|)}(t^*_{B_j(1)} + p_{B_j(1)B_j(|B_j|-1)})) - \lambda^*_{B_j(|B_j|)-1} & \\ \textbf{iv)} & \lambda^*_i \geq 0 & \forall i \in [\![B_j(1), B_j(|B_j|)]\!] \end{cases} \tag{72}$$

Necessary condition proof: We refer to p_{ij} for $(i \leq j) \in [\![1, n]\!]^2$, as the cumulative processing duration of activities a_i to a_j. Relying on Proposition (2), we can sum i) , ii) and iii) of (72), leading to:

$$0 \in \delta c_{B_j(1)}(t^*_{B_j(1)}) + \sum_{i=B_j(1)+1}^{B_j(|B_j|)} \delta c_i(t^*_{B_j(1)} + p_{B(1)i-1}) \Rightarrow 0 \in \delta C_{B_j}(t^*_{B_j(1)}) \tag{73}$$

and thus, following the definition of the optimal block execution cost of Equation (18), $t^*_{B_i(1)} \in [T^{-*}_{B_i}, T^{\dashv *}_{B_i}]$. Any optimal solution thus verifies the first statement of Theorem (1). Finally, for any block B_j and prefix block B^k_j, summing i) , ii) and iii) for $j \in [\![B_j(1)+1, k]\!]$ leads to:

$$-\lambda^*_k \in \delta c_{B_j(1)}(t^*_{B_j(1)}) + \sum_{i=B_j(1)+1}^{k} \delta c_i(t^*_{B_j(1)} + p_{B(1)i-1}) \Rightarrow -\lambda^*_k \in \delta C_{B_j}(t^*_{B^k_j(1)}) \tag{74}$$

which can be reformulated as $T^{\dashv *}_{B^k_j} \geq t^*_{B_j(1)}$ and implicates the last statement of Theorem (1).

Sufficient condition proof: Supposing that a solution $\mathbf{t} = (t_1 \dots t_n)$ is given with its blocks $(B_1 \dots B_m)$, respecting conditions of Theorem (1). Following block definitions and primal feasibility, it only remains to prove that Conditions (72) are respected for each block. We

choose the following Lagrangian multipliers, which are non negative as $T^{t*}_{B^k_i} \geq t_{B_i(1)} \Rightarrow \exists x \leq 0 \in \delta C_{B^i_j}(t_{B_j(1)}))$.

$$\forall j \in [\![1, m]\!], \begin{cases} \lambda^*_i = -\min(x \in \delta C_{B^i_j}(t_{B_j(1)})) & \forall i \in [\![B_j(1)+1, B_j(|B_j|)-1]\!] \\ \lambda^*_{B_j(|B_j|)} = 0 \end{cases} \tag{75}$$

Proposition (2) then involves that for any $j \in [\![1, m]\!]$ and $i \in [\![B_j(1)+1, B_j(|B_j|)-1]\!]$,

$$-\lambda^*_i \in \delta C_{B^i_j}(t_{B_j(1)}) = \delta c_{B_j(1)}(t_{B_j(1)}) + \sum_{k=B_j(1)+1}^{i} \delta c_k(t_{B_j(1)} + p_{B_j(1)k-1}) \tag{76}$$

In the case where $i = B_j(1)$, Equation (76) proves statement i) of (72). Also, $\lambda^*_{B_j(1)} \in \delta(-c_{B_j(1)})(t_{B_j(1)})$, and we can sum this statement with Equation (76) for $i = B_j(1)+1$, using Proposition 2), and leading to:

$$\begin{aligned} \lambda^*_{B_j(1)+1} - \lambda^*_{B_j(1)} \in \delta(-c_{B_j(1)})(t_{B_j(1)}) + \delta c_{B_j(1)}(t_{B_j(1)}) + \delta c_{B_j(1)+1}(t_{B_j(1)} + p_{B_j(1)}) \\ = \delta c_{B_j(1)+1}(t_{B_j(1)} + p_{B_j(1)}) \end{aligned} \tag{77}$$

The remaining statements of Equation ii), and Equation iii) in (72) are proven by recurrence, assuming that for a given $i \in [\![B_j(1)+1, B_j(|B_j|)-1]\!]$, $\lambda^*_{i-1} - \lambda^*_i \in \delta c_i(t_{B_j(1)} + p_{B_j(1)i})$ leads to:

$$\begin{aligned} \lambda^*_i - \lambda^*_{i+1} &= -\lambda^*_{i+1} + \lambda^*_{i-1} - (\lambda^*_{i-1} - \lambda^*_i) \\ &\in \delta c_{B_j(1)}(t_{B_j(1)}) + \sum_{k=B_j(1)}^{i+1} \delta c_k(t_{B_j(1)} + p_{B_j(1)k-1}) + \delta(-c_{B_j(1)})(t_{B_j(1)}) \\ &+ \sum_{k=B_j(1)}^{i-1} \delta(-c_k)(t_{B_j(1)} + p_{B_j(1)k-1}) + \delta(-c_i)(t_{B_j(1)} + p_{B_j(1)i-1}) \\ &\subset \delta c_{i+1}(t_{B_j(1)} + p_{B_j(1)i}) \end{aligned} \tag{78}$$

All the sufficient optimality conditions of the problem are thus satisfied by solution $\mathbf{t} = (t_1 \dots t_n)$. □

C Re-optimization and time-dependent processing times

The original time-dependent problem presented by Hashimoto et al. (2008) in VRP context involves the determination of both services dates to customers and departure dates on the route. Both services and travels are considered here indifferently as activities characterized by a time-dependent cost, and a time and sequence dependent processing time. This problem can then be reformulated as $\{\Sigma c_i(\Delta t_i), P(t)\}$, and is given in Equations (79-80), where function $c_{ij}(t)$ are assumed to take infinite value for $t < 0$.

$$\min_{t_1 \dots t_n} \quad \sum_{i=1}^{n} c_{\sigma_i \sigma_j}(t_{\sigma_i}) \tag{79}$$

$$s.t. \quad t_{\sigma(i)} + p_{\sigma(i)\sigma(j)}(t_{\sigma(i)}) \leq t_{\sigma(i+1)} \qquad 1 \leq i < n \tag{80}$$

When the functions are piecewise linear, lower semicontinuous, non negative, and that property (HYI) (Section 6.4) is satisfied, the following re-optimization approach can be applied.

Data at play: Optimal cost $F(A_i)(t)$ (respectively $B(A_i)(t)$) of a schedule for A_i, when the first (the last) activity is started before (after) t.

Data computation: For a sequence A with a single activity, $F(A)(t) = \min_{0 \leq x \leq t} c_{A(1)}(x)$ and $B(A)(t) = \min_{x \geq t} c_{A(1)}(x)$. The forward and backward dynamic programming equations of (81-82) can then be used to compute functions $F(A_i)(t)$ and $B(A_i)(t)$ respectively on prefix and suffix subsequences of the incumbent timing problem:

$$F(A_i \oplus A)(t) = \min_{0 \leq x \leq t} \{c_{A(1)}(x) + \min_{x' : x' + p_{A_i(|A_i|)A(1)}(x') \leq x} F(A_i)(x')\} \tag{81}$$

$$B(A \oplus A_i)(t) = \min_{x \geq t} \{c_{A(1)}(x) + \min_{x' \geq x} B(A_i)(x' + p_{A(1)A_i(|A_i|)}(x'))\} \tag{82}$$

Evaluate concatenation: Equation (83) returns the optimal cost $Z^*(A_1 \oplus A_2)$ of the timing problem related to the concatenation of a prefix and a suffix subsequence.

$$Z^*(A_1 \oplus A_2) = \min_{0 \leq x} \{ \min_{x' : x' + p_{A_1(|A_1|)A_2(1)}(x') \leq x} F(A_1)(x') + B(A_2)(x)\} \tag{83}$$

Evaluating the concatenation of two subsequences A_1 and A_2 can be performed in $O(\varphi_c + \varphi_p)$, where φ_c and φ_p denote respectively the total number of pieces in the cost functions and time-dependent processing times functions of the subproblem at play. Yet, this approach actually enables only to concatenate a pair of subsequences. For series of timing problems issued from more than two sequences, as in many neighborhood searches for TSP or VRP for example, additional properties on the order of evaluation of the timing problems are necessary to lead to efficient re-optimization approaches.